SIMARGL

D 2.4 Design for Secure, Fair and Explainable Machine Learning

Work Package 2: Architecture Specification of the SIMARGL Toolkit to Detect and Counter Malware and Stegomalware

Document Dissemination Level

P	Public	☒
CO	Confidential, only for members of the Consortium (including the Commission Services)	☐

Document Due Date: 30/11/2019
Document Submission Date: 29/11/2019

This work is performed within the SIMARGL Project – Secure Intelligent Methods for Advanced RecoGnition of malware and stegomalware – with the support of the European Commission and the Horizon 2020 Program, under Grant Agreement No 833042

Document Information

Deliverable number:	**2.4**
Deliverable title:	Design for Secure, Fair and Explainable Machine Learning
Deliverable version:	0.6
Work Package number:	2
Work Package title:	Architecture Specification of the SIMARGL Toolkit to Detect and Counter Malware and Stegomalware
Due Date of delivery:	30/11/2019
Actual date of delivery:	29/11/2019
Dissemination level:	P
Editor(s):	Pluribus (deliverable responsible)
Contributor(s):	Pluribus (deliverable responsible) FUH (Beneficiary) ITTI (Beneficiary)
Reviewer(s):	CNR (Beneficiary) WUT (Beneficiary)
Project name:	Secure Intelligent Methods for Advanced RecoGnition of malware and stegomalware
Project Acronym	SIMARGL
Project starting date:	1/5/2019
Project duration:	36 months
Rights:	SIMARGL Consortium

Version History

Version	Date	Beneficiary	Description
0.1	17.09.2019	Pluribus	Initial version
0.2	02.10.2019	Pluribus	Added more references, and improved writing/organization of contents according to partners' feedback
0.3	12.11.2019	ITTI, FUH	State of the art on SECURITY, FAIRNESS and EXPLAINABILITY of ML
0.4	18.11.2019	Pluribus	First complete version. Evaluation of technical products, and possible integration scheme
0.5	22.11.2019	CNR, WUT	Changes in format and other minor corrections introduced during the review
0.6	26.11.2019	Pluribus	Final review

Abbreviations and Acronyms

ACRONYM	EXPLANATION
ML	Machine Learning
AI	Artificial Intelligence
LIME	Local Interpretable Model-Agnostic Explanations
MMD	Maximum Mean Discrepancy
GDPR	European General Data Protection Regulation 2016/679
DeepLIFT	Deep Learning Important Features
BP	Backpropagation
PCA	Principal Component Analysis (PCA)
t-SNE	t-Distributed Stochastic Neighbor Embedding
PDF	Portable Document Format
DNS	Domain Name System

Table of Contents

Table of Figures

Table of Tables

1. Introduction

Machine Learning (ML) algorithms are nowadays recognized as mandatory to cope with the increasing variability and sophistication of computer security threats. The key feature of ML approaches is their ability to *generalize* from a given set of examples (training data, e.g., data associated to legitimate and/or malicious activities) and thus potentially detect never-before-seen (zero-day) attacks or variants of known ones.

In the context of computer security, *any* solution taking advantage of ML techniques is expected to deliver:

- **Accuracy**. ML should be able to provide accurate classification results, i.e., accurately distinguish data associated to computer security threats from data associated to legitimate activities. This property is fundamental to allow security operators to focus on events that are most likely associated with security threats.
- **Adaptiveness**. ML algorithms should be capable to quickly adapt their parameters taking into account new data about security threats/legitimate instances. This property is fundamental to deal with the *rapid evolution* of threats. Adaptability may greatly depend on the data used to learn the system as well as on the knowledge of defenders (capability to correctly *model* novel threats).

These two properties intrinsically reflect the reason why ML algorithms are employed in such context and thus constitute a *baseline*. On the other hand, when applied to computer security, ML algorithms should also deliver[1]:

- **SECURITY (Robustness).** Accuracy should be guaranteed in the presence of attack variations, never-before-seen attacks, or even deliberate attacks against the ML system itself. This property is fundamental to deal with the *adversarial evolution* of computer security threats (arms-race between attackers and defenders).
- **FAIRNESS**. ML should be capable to deal with *biased* training data. Biased training data is dangerous because it may mislead the learning algorithm and thus any human decision based on its output.
- **EXPLAINABILITY**. ML algorithms should provide explainable results, i.e., results that can be easily understood by security operators, to take informed decisions. This property is fundamental to allow humans to be profitably in the loop, because systems alone may not deal effectively with human adversaries (attackers).

The goal of this deliverable is to provide guidelines for the design of ML systems capable to *satisfy* all such properties. These guidelines can be exploited by technology providers in the SIMARGL consortium for the improvement of their ML tools as well as for the development of ML components in the whole SIMARGL platform.

The rest of the document is organized as follows.

- In **Section 2**, we present an overview of state-of-the-art work related to the security, fairness and explainability of ML systems.
- In **Section 3** we propose a general framework for the evaluation of different design choices according the above-mentioned, desired properties of ML systems. We show that design components may be "intelligently" combined to achieve a superior tradeoff between these properties, exploiting the complementarity of each single design choice.
- In **Section 4**, we apply the proposed framework in a realistic case study on web attack detection.
- Finally, in **Section 5**, we apply our framework for the evaluation of each ML tool developed by technology providers in SIMARGL and provide detailed suggestions for their improvement. In such section, we also present a possible way to integrate SIMARGL tools according to such evaluation.

[1] Indeed, there are many other desired properties, but in this deliverable, we will focus on these ones. For instance, important properties that are not considered here are *learning time* (time required to learn the system/adapt its parameters) and *classification throughput* (e.g., time required to extract and classify each data sample).

2. Current research lines in secure, fair and explainable Machine Learning

2.1 Dealing with an adversary

A key point when designing ML systems is to recognize that they should be used by humans (e.g., security operators) to protect against other humans (e.g., cyber criminals). However, ML algorithms have been designed under the assumption that training and test data follow the same underlying probability distribution. This means that the algorithms themselves may be threatened by skilled attackers which deliberately violate this assumption. In particular, attackers may be highly motivated[2] to:

- manipulate data at test time to evade detection (**Evasion** attack);
- and/or inject poisoning samples (**Poisoning** attack) into the training data to mislead the learning algorithm and subsequently cause misclassifications (at test time).

Learning from data alone is clearly not sufficient to achieve ML **security** (robustness). We need to ask *what-if* questions to imagine how data can evolve/modify due to threat specifics or deliberate adversarial attacks. In other terms, we need to develop the ML system according to an expert-driven, adversary-aware model that accurately takes into account the *specifics of the threat* being detected as well as *how* adversaries may react and the related *consequences* of their actions on the ML system.

2.2 Fairness in ML/AI

As the fairness in ML/AI is a trending topic, there are many algorithms focusing on improving fairness in ML described in the literature. Most of them fall into three categories: preprocessing, optimization at training time, and post-processing [1]. In general, algorithms belonging to the same category are characterized by common advantages and flaws. These three categories are characterized below.

2.2.1 Preprocessing

The idea is based on building a new representation of input data by removing the information correlated to the sensitive attribute and at the same time preserve the remaining input information as much as possible. The downstream task (e.g., classification, regression, and ranking) can thus use the "cleaned" data representation and produce results that preserve demographic parity and individual fairness.

In [2] authors use the optimal transport theory to remove disparate impact of input data. They also provide numerical analysis of the database fair correction. In [3] authors propose a learning algorithm for fair classification addressing both group fairness and individual fairness by obfuscating information about membership in the protected group. Authors of [4] propose a model based on a variational autoencoding architecture with priors that encourage independence between sensitive and latent factors of data variation. To remove any remaining dependencies an additional penalty term based on the Maximum Mean Discrepancy (MMD) measure is additionally introduced. A statistical framework for removing information about a protected variable from a dataset is presented in [5], along with the practical application to a real-life dataset of recidivism, proving successful predictions independent of the protected variable, with the predictive accuracy preserved. Reference [6] proposes a convex optimization for learning a data transformation with three goals: controlling discrimination, limiting distortion in individual data samples, and preserving utility.

The advantages that are common for fair pre-processing algorithms include the possibility to use preprocessed data for any downstream task and no need to modify the classifier. There is also no need to

[2] Cyberattacks allow for high strategic gains (financial, political, etc.) at a global scale with very low risks and costs compared to physical actions. Threat actors may be thus highly motivated to bypass security mechanisms, and stealthily execute unauthorized operations to hide their traces.

access the sensitive attributes at testing time. In contrast, preprocessing algorithms can be used only to preserve statistical parity and individual fairness, and their performance in terms of accuracy and fairness trade-off are not as promising as in the case of two other groups of algorithms.

2.2.2 Optimization at training time

Data processing at training time provides good performance on accuracy and fairness measure and ensures higher flexibility in optimizing the trade-off between these factors. The author of [1] describes that the common idea that can be found in the state-of-the-art works falling into this category of algorithms is to add a constraint or a regularization term to the existing optimization objective. Recent works considering algorithms to ensure ML fairness applied at the training time include: [7] where the problem of learning a non-discriminatory predictor from a finite training set is studied to preserve "equalized odds" fairness, [8] and [9] where a flexible mechanism to design fair classifiers by leveraging a novel intuitive measure of the decision boundary (un)fairness is introduced, and [10] that addresses the problem of reducing the fair classification to a sequence of cost-sensitive classification problems, whose solutions provide a randomized classifier with the lowest (empirical) error subject to the desired constraints.

The disadvantages of the abovementioned approaches include the fact that these methods are highly task-specific and they require a modification of the classifier, which can be problematic in most applications/cases.

2.2.3 Post-processing

The post-processing algorithms are focused on editing the outcome to satisfy the fairness constraints and can be applied to optimize most of fairness definitions except the counterfactual fairness. The basic idea is to find a proper threshold using the original score function for each group. An exemplary recent work that falls into this category is publication [11], in which the authors show how to optimally adjust any learned predictor to remove the discrimination according to the "equal opportunity" definition of fairness, with the assumption that data about the predictor, target, and membership in the protected group are available.

The advantage of post-processing mechanisms is that retraining/changes are not needed for the classifier (the algorithm can be applied after any classifier). Another benefit comes in the form of good performance in the terms of fairness measures. The disadvantages include a need for test-time access to the protected attribute and the lack of flexibility in picking the accuracy–fairness tradeoff [1].

2.2.4 Summary

The author of [10] provides an experimental comparison of the selected algorithms applied to reduce unfairness using four real-life datasets with one or two protected sensitive attributes (gender or/and race). The selected methods include preprocessing, optimization at training time and post-processing approaches. The methods that achieve the best trade-off between accuracy and fairness are those falling into the optimization at training time category, while the advantage related to the implementation of preprocessing and post-processing methods is the preservation of fairness without modifying the classifiers. In general, experimental results prove the ability to significantly reduce or remove the disparity, in general not impacting the classifier's accuracy for all the methods. The reduction methods (optimization at training time) used to preserve the demographic parity achieve the lowest constraint violations, outperforming or matching the baselines. The post-processing algorithm performs well for small violations. Pre-processing algorithms (based on reweighting and relabeling) achieve the worst fairness measures [1][10].

2.3 Explainability and interpretability in ML/AI

The aspects of explainability and interpretability are trending topics in the area of ML and Artificial Intelligence (AI) in general as well. As discussed in [12] and [13], explainability and interpretability tend to be used (also in literature) as interchangeable terms, however despite the fact that they are related concepts, there are some minor differences in their meanings. Interpretability addresses aspects related to observation

of AI system outputs. Interpretability of AI system is higher, if the changes of the systems outputs in result of changing algorithmic parameters are more predictable. In other words, system interpretability is related to the extent to which a human can predict the results of AI systems based on different inputs. In contrast, explainability is related to the extent to which a human can understand and explain (literally) the internal mechanics of an AI/ML system. In its simplest form, the definition of explainability refers to an attempt to provide insights into the predictor's behaviour [14].

According to [13], nowadays, attempts to define these concepts are not sufficient to form a common and monolithic definition of explainability and interpretability and to enable their formalization. It is also worth to mention that, the "right to explanation" in the context of AI systems directly affecting individuals by their decisions, especially legally and financially, is one of the subjects of the GDPR [15].

Different scientific and literary sources focus on surveying and providing a categorization of methods and techniques addressing explainability and interpretability of decisions provided by the use of AI systems. Reference [12] discusses the most common practical approaches, techniques and methods used to improve ML interpretability and to enable more explainable AI. They include, among the others, algorithmic generalization, i.e., shifting an attitude from case-specific models to more general ones. Another approach is paying attention to feature importance, described also in [14] as the most popular technique addressing ML explainability, also known as feature-level interpretations, feature attributions or saliency maps. Some of feature importance-based methods found in the literature are perturbation-based methods based on Shapley values adapted from the cooperative game theory. In the explainability case, Shapley values are used to attain fair distribution of gains between players, where a cooperative game is defined between the features. In addition, some recent works [12][16] show that adversarially trained models can be characterized by increased robustness but provide also clearer feature importance scores, contributing to improved prediction explainability. Similar to the feature importance way, counterfactual explanations [17] is a technique applied in the financial and healthcare domains. Explanations using this method are based on providing point(s) and values that are close to the input values for which the decision of the classifier possibly changes (case-specific threshold values). Another method used for increasing explainability of AI-based predictions is Local Interpretable Model-Agnostic Explanations (LIME) based on approximation of the model by testing it, then applying changes to the model and analysis of the output. Deep Learning Important Features (DeepLIFT) model is used for the challenge-based analysis of deep learning/neural networks. As described in [12] DeepLIFT method is based on back propagation (BP), i.e., digging back into the feature selection inside the algorithm and "reading" neurons at subsequent layers of network.

In the literature one can find different attempts of categorization of the methods aimed at increased explainability of AI. Integrated/intrinsic and post-hoc explainability methods [13][17] are examples of such categorizations. Intrinsic explainability in its simplest form is applicable to the low complexity models (linear ones, decision trees, and rule-based) where the explanation of a simple model is the model itself. On the other hand, more complex models are explainable in a post-hoc way, providing explanations after the decision and using techniques such as feature importance, layer-wise relevance propagation, or mentioned Shapley values. Other forms of post-hoc explanations include also textual and visual justification of the decision.

Similar categorization is given in [17] where in-model (integrated/intrinsic) and post-model (post-hoc) methods exist alongside additional pre-model interpretability methods. Pre-model methods are applicable before building (or selection) of the ML model and are strictly related to the input data interpretability. They use mainly classic descriptive statistical methods, such as Principal Component Analysis (PCA), t-Distributed Stochastic Neighbor Embedding (t-SNE), and clustering methods such as *k*-means. Another criterion described in [17] is the differentiation into model-specific and model-agnostic explanation methods. In the majority of cases model-specific explanation methods are applicable to the intrinsically interpretable models (for example, analysis and interpretation of weights in a linear model), while model-agnostic methods can be

applied after the model and include all post-hoc methods relying on the analysis of pairs of feature input and output.

Alternative criterion based on explanation methods is described in [18]. In such differentiation, methods are categorized based on type of explanation that given method provides, including: feature summary (providing statistic summary for each feature with their possible visualization), model internals (for intrinsic explainable or self-explainable models), data point (example-based models) and surrogate intrinsically interpretable model – that is trained to approximate the predictions of a black box model.

According to [17] and [19], explanation models can be evaluated and compared using qualitative and quantitative metrics, as well as by comparison of the explanation method's properties, including its expressive power, translucency (model-specific vs. model-agnostic), portability (range of applications) and computational complexity. On the other hand, individual explanations can be characterized by accuracy, fidelity, consistency (similarity of explanations provided by different models), stability, comprehensibility, certainty, to list most relevant ones. According to the literature we can also distinguish qualitative and quantitative indicators to assess the explanation models. Factors related to quality of explainability are: form of the explanation, number of the basic units of explanation that it contains, compositionality (organization and structure of the explanation), interactions between the basic explanation units (i.e., intuitiveness of relation between them), uncertainty and stochasticity. Quantitative indicators are presented in some works (see, e.g., [17][20][21]). The most common metrics used to quantify the interpretation of ML models are identity, separability and stability. These three factors provide the information on to what extent identical, non-identical and similar instances of predictions are explained in identical, non-identical and similar way, respectively. In addition, according to [20] the explanation should be characterized by high completeness (coverage of the explanation), correctness and compactness. However, these indicators are applicable only to simple models (rule-based, example-based).

3. Proposed Framework

3.1 Risk management problem

ML properties described in Section 1 can be guaranteed up to a certain degree (likelihood). And their violation may lead to unexpected results and failures of the ML system (impact). In other terms, there is an underlying *risk* (impact x likelihood) associated to such properties. Thus, we propose to approach the problem as a *risk management problem*. Please note that computer security is *intrinsically* a risk management problem: our idea is to apply the same approach to the evaluation of different ML design choices. Additionally, the results of this evaluation can be readily applied for the evaluation of the whole SIMARGL platform. Using such approach, the goal is to

(1) **evaluate** the risk (likelihood x impact) associated to each violation of the above-mentioned properties,
(2) identify suitable design choices that **mitigate** the underlying risk so that it becomes *acceptable*.

This is clearly an iterative process, where desired properties may be *inter-dependent*. In particular, we propose to evaluate/mitigate the risk associated to each property at different abstraction levels. For the scope of our discussion, we can refer to three fundamental levels commonly used in pattern recognition, namely:

1) Data collected (**Data**): Section 3.1.1.
2) Features extracted from data (**Feature**): Section 3.1.2.
3) ML Algorithm used to classify the extracted features (**Classifier**): Section 3.1.3.

Depending on the ML task, mitigation measures may focus on one of these phases to effectively reduce the risk. Nonetheless, this choice must be *always* the output of a risk evaluation process in *all* phases. This is because the risk associated to lower abstraction levels (e.g., Data) may unavoidably be transferred to higher abstraction levels (e.g., Classification).

3.1.1 Data Level

Fundamental aspects of data are its *target class* and its *source*. Depending on such aspects, we may achieve different tradeoffs between the desired properties described in Section 1. In the following, we present the typical (high-level) options that one may encounter in practice.

3.1.1.1 Target Class

The **target class** of data, i.e., the category of activities that the data is expected to describe, may be either *attack* (threat) or *legitimate*. In the former case (attack class), we may achieve maximum ***explainability***, because data may be associated to a known threat, for which may be available an accurate description of goals, and exploited vulnerabilities (e.g., CVE[3]). On the other hand, learning from such data allows at best to model (and thus detect some variations of) known attacks. In this case, the ***accuracy*** of the system may be essentially limited to the detection of known attacks. In turn, this means also ***low robustness*** against an adversary that intelligently modify attack data (e.g., a known attack) to ***evade*** the system.

On the other hand, in the latter case (legitimate class), ***explainability*** may be limited to the description of legitimate activities, because data tells nothing about attacks as security operators would like. However, by means of an anomaly-based approach (detection of data which deviates from the collected data), the system may ***accurately*** detect any kind of attack, both known and unknown (*zero-day*). This potentially allows to detect any kind of adversarial attack as anomalous data (***robustness against evasion***).

[3] https://cve.mitre.org/

3.1.1.2 Source of data

The **source of data** is very important to determine the exposition of the system to ***poisoning*** attacks and the ***adaptability*** of the system. The key question is whether and how an adversary may take control of the data we are using for learning the system. First of all, note that ***data at test time may be unavoidably affected/modified by an adversary*** because the detection system is expected to work on an adversarial environment in which its task is to distinguish malicious activities from legitimate ones. Hence, the source of data at test time may be either the attacker or legitimate users.

On the other hand, the source of data used for learning should be thoroughly vetted for potential poisoning vulnerabilities. A general trade-off can be observed between robustness against poisoning vulnerabilities and adaptability. This is because accurate data labelling (e.g., legitimate/malicious) requires a significant effort for human operators, and takes time, thus it limits the adaptability of the system. However, it protects against poisoning attacks, because data may be thoroughly vetted by security experts and adversarial noise may be more limited/absent.

For example, let us assume to acquire our data from a public repository such as *exploit db*[4], manually selecting only working exploits. It is highly unlikely that such data will contain poisoning noise against our algorithm. On the other hand, adaptability (as well as representativity/accuracy) will be substantially limited to the collected and validated exploits.

High ***adaptability*** can be supported by the continuous (real-time) acquisition of data, assuming that it represents a certain class (e.g., legitimate/malicious). In this case, data cannot be thoroughly vetted by security experts so (adversarial) noise is more likely.

For instance[5], let us assume that our source of data is automatically gathered from **VirusTotal**[6], the popular online system capable to analyze a file using multiple antivirus/antimalware systems. Let us also assume to define a simple heuristic (used in many research papers): a file is considered as malicious if flagged as malware by at least a certain percentage of antivirus systems (say, e.g., 50%). We may *automatically* collect our malware dataset from VirusTotal using a simple search query (download all files flagged by at least 50% of antivirus systems). This simplicity hides a significant ***poisoning vulnerability***. An adversary may submit an ***arbitrary*** number of samples against VirusTotal (submissions are free and do not require authentication), such that they are flagged as malicious by at least 50% of antivirus. This is quite easy to do, e.g., embedding code that matches with known malware signatures[7]. Such samples, however, may be ***totally benign***[8] and also embed poisoning data for our learning algorithm. In other terms, such samples may not constitute a harm for users, but specifically for our learning system.

It is important to note that an adversary-aware choice of our data source may significantly strengthen the robustness of our ML system against poisoning attacks, increasing the ***cost per adversarial sample*** that the adversary must incur when injecting a malicious sample into the data used for learning.

[4] https://www.exploit-db.com/

[5] This scenario is indeed feasible, even if, to the best of our knowledge, no previous work demonstrated it experimentally. It is very useful to explain the threat in a concrete case.

[6] https://www.virustotal.com

[7] This technique is also known as *overstimulation.* A number of research works demonstrated the feasibility of such attacks against network intrusion detection. See for instance, one of the first works:
S. Patton, W. Yurcik, and D. Doss, "An achilles' heel in signature-based ids: Squealing false positives in Snort," in Proceedings of Fourth International Symposium on Recent Advances in Intrusion Detection, vol. 10, October 2001, p. 12.
An analogous technique can be used against antivirus systems and thus VirusTotal.

[8] They may not constitute a harm for users, they are just crafted to match with known signatures.

For instance, we can modify the query used to retrieve our dataset, selecting only a limited number of files (e.g., one) per authenticated user in VirusTotal. In this scenario, the ***data source becomes each single authenticated user*** in VirusTotal. Thus, to inject poisoning noise into our system, the adversary must submit its files as authenticated user and the number of poisoning samples will strictly depend on the number of accounts he/she is able to register on the platform. This operation may increase the cost of poisoning noise for the adversary, thus decreasing the likelihood of adversarial attacks. This comes also at the price of a reduced dataset (potentially less representative), and thus accuracy/adaptiveness of the learned detection system for the defender.

Another relevant example is related to a research paper published in a top Computer Security Conference (IEEE Symposium on Security and Privacy). In that paper, the authors proposed a ML system for the automatic generation of signatures for polymorphic worms [22].

A key vulnerability of the system was related to the **source of data used for training**, i.e., the worm itself, in real-time. Please note that this is an extreme case in which **all training data is under the control of the adversary**. In fact, the worm itself could deliberately and easily include poisoning noise against the ML learning system. This vulnerability was in fact exploited one year later, by other researchers (paper published in the same top conference [23]), and by the same authors in another relevant conference (Recent Advances in Intrusion Detection [24]), to successfully mislead the ML system. Such example clearly highlights a critical point. If an adversarial evaluation is absent in the design of detection tools, their validity/reliability will be rather weak (in this case, the approach validity "lasted" just one year).

3.1.1.3 Summary of main tradeoffs in data acquisition

Target Class	**Pros**
Attack	Explainability
Legitimate	Accuracy, Robustness (against evasion)

Data source	**Pros**
Real-time	Adaptability
Manually-vetted	Robustness (against poisoning)

3.1.2 Feature Level

The features (measurements) extracted from data must reflect *invariant* aspects of the activities you want to classify. In this way, the adversary should incur into (significant) additional efforts and costs to generate data that causes classification errors (e.g., evade or overstimulate the classifier).

Such features are also referred to as ***conserved*** features, and defined to as "features that cannot be unilaterally modified without compromising malicious functionality" in [25].
The above-mentioned publication proposed an empirical approach for the identification of such features in the context of PDF malware, using off-the-shelf manipulation tools. The idea is to manipulate known malware to understand what features are conserved (invariant) to such manipulations, and to what extent. This approach is interesting and has a practical validity. However, it must be noted that conserved features identified with such method may strictly depend on known file manipulations, and in particular, those offered by the employed tools. Thus, in some sense, the result is limited to what manipulations are encompassed by such tools: it may thus lack ***fairness*** (results biased on known/off-the-shelf manipulations) as well as ***explainability*** (why such features are conserved? Do they really reflect invariant aspects of attacks?).

While this is an area of active research, we observe that a general approach to such adversarial problem can be based on *causal reasoning* advocated by the 2011 A.M. Turing Award winner, Judea Pearl[9] [26].

Let us define a generic set of measurements (pattern) with $X=\{x_1, \ldots, x_n\}$ and the outcome class as Y (Y=0 legitimate, Y=1 attack). The concept of *invariant* can be expressed as a ***counterfactual***:

- an attack (Y=1) would not have happened if X was not observed

The key concept is that **expert domain knowledge plays a central role** to both identify candidate invariant features/patterns (**security**) and to *motivate* such choices. The motivation behind the choice of features should also be associated with their *meaning*, so it may be beneficial for the **explainability** of detections. Interestingly, causal modelling can also effectively deal with *biased* training data and thus achieve ML **fairness**, as shown in this recent work [27].

To clarify these concepts let us give a concrete example related to PDF malware detection. A number of research works presented *excellent* detection result using features related to the **PDF document structure**. PDF malware in the wild may be characterized by significant structural differences with respect to legitimate PDF, and this explains why the experiments performed using PDF structural features showed an *impressive* accuracy. See for instance two relevant papers published in top security conferences [28] [29].

However, the structure of a PDF document **is not by any means an intrinsic feature of PDF malware studied in such works**. It was instead a ***bias*** in PDF malware collected in the wild, caused by the way in which the exploits were implemented. This means that the learned classifier and the obtained classification results were **not fair** (heavily biased on document structure). Figure 1 shows the causal model for such malware detection problem. The measured variable was the PDF structure (Z). However, as it can be seen from the diagram, Z→Y does not hold (the structure of PDFs X does not *cause* an attack/legitimate instance)[10]. The true causal relationship is between X (executed code)→Y.

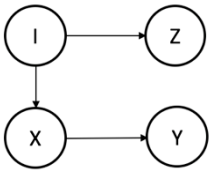

Figure 1. Causal diagram for PDF malware detection. I=attack implementation, X=intrinsic pattern (e.g., malicious JavaScript instructions), Z=measured pattern (PDF structure), Y=instance class (attack/legitimate)

In fact, subsequent research work [30] showed that an adversary may **easily change the attack implementation**, to deliver the same attack pattern X with a PDF file whose structure Z is typically observed in legitimate PDFs.

This example clearly highlights that the concept of intrinsic features is an aspect not completely clear/recognized by researchers in computer security, even for those who published in top conferences (and the related reviewers). However, it is clear that we must design systems that look at aspects (features) that

[9] https://amturing.acm.org/award_winners/pearl_2658896.cfm

[10] Indeed, there may be also attacks exploiting a *malformed* structure. In this case, the PDF structure may be the cause of the attack, and the causal relationship Z→Y may hold (so, there may be an arrow between Z and Y). We also observed some of these attack instances in the wild, where a malformed structure was deliberately aimed at **evading parsing/analysis mechanisms** (attacks against the PDF malware detector itself). However, this is a completely different class of attacks compared to that studied by the referenced research works.

are difficult/costly to be changed by adversaries. On the other hand, the main goal of computer security is to increase the effort for an adversary up to a level which is not more convenient for her/him to reach her/his goals.

Staying in the context of PDF documents, an intrinsic feature of most PDF malware in the wild is associated to the execution of malicious code X, and in particular *JavaScript*. Thus, a more robust approach to fight malware in the wild is to focus on the analysis of JavaScript code embedded in PDF files, as done in [31].

That paper also proposed a ML algorithm capable to take into account possible adversarial modifications of the extracted features. Please note that in this case pattern X is related to the JavaScript instructions of PDF malware. So, the proposal is valid only if the assumed causal model X→Y is valid. **Thus, understanding the assumed causal (adversarial) model is also useful to clearly frame the limits of the approach**. ML results are ***fair*** when the causal model X→Y is valid.

A very interesting aspect is that extracting intrinsic features allows one to deal with vulnerabilities at data level. This is especially true for **data passively acquired in large computer networks**. If features capture intrinsic aspects of the attack, modifying the attack to evade detection or inject poisoning patterns may weaken the attack itself against the network users (counterproductive for the adversary). A clear example of this case is in [32], where a set of intrinsic features of fast-flux networks was used. A discriminant feature was the *novelty, i.e.*, the fraction of new IP addresses observed for the botnet across the time. If the attacker tries to change this feature to evade the system, it has to rely on a rather stable set of (reliable) IP addresses. This is because it *must still deliver successfully the attack against the network users*. But if this set is rather stable, it can also be blacklisted easily by defenders to fight the phenomenon. This behaviour would be *counterproductive* for an adversary, because it would be against the actual proposal of fast-flux networks, i.e., blacklist evasion and robustness of malicious content delivery.

3.1.3 Classifier Level

In general, we can identify three main classification techniques:

1. **One-class attack**. This classifier learns from attacks. It aims to classify patterns as malicious if they agree with a previously learned model of attacks. The pro of this model is in the **explainability**, especially if extended to multi-class, i.e., **one different class per each attack** for which we have a description.
2. **One-class legitimate**. This classifier learns from legitimate activities, and it aims at classifying patterns as malicious (anomalous) if they do not agree with the learned model of legitimate activities. The pro of this model is in the **robustness** and **accuracy**, if intrinsic features are selected, because it may detect both known and unknown attacks (zero-days) or even attempts of evasion by a skilled attacker.
3. **Two-class**. This classifier learns from both legitimate activities and attacks to build a *discriminant* classification model. The main pro of this model is its ability to exploit discriminant information about the two classes to attain **superior classification accuracy** with respect to previous two techniques.

Please note that in general, multiple, independent classifiers should be used. This is a suggested option to deal with different classes of attacks (**each classifier specialized on a specific class of attack, i.e., multi-class attack**), because it allows to deal with the complexity of the attack detection task and allows to satisfy the explainability requirement described in Section **Errore. L'origine riferimento non è stata trovata.**. In the case of one-class legitimate classifiers, it is thus suggested to develop specialized anomaly detection models, whose output (anomalous values) can be easily interpreted by a security operator and highlight a general class of attacks. We will provide a concrete example in Section 3.1.4.

3.1.4 Intelligent, adversary-aware combination of design choices

Each data acquisition (Section 3.1.1) and classification (Section 3.1.3) approach has different pros and cons. Recognizing this aspect allows us to understand that **only a combination of these approaches** can yield

interesting tradeoffs between the five desired features described in Section 1. In Figure 2, we outline a general combination that as a whole allows to join the pros and narrow down the cons of each design choice.

As it can be seen, we acquire data whose target is both attacks and legitimate instances. Real-time data acquisition allows for **high adaptability** of a one-class model of legitimate activities. Such model is learned using a set of intrinsic features (pattern) for the threat being detected. Such features allow one to accurately discriminate attack patterns as *outliers* from data, so it may detect either known or zero-day attacks (**high accuracy**). Anomalous patterns may also be the input for a multi-class classifier, learned on different attack instances. This way, anomalies may be either classified as a known attack (if recognized by the attack classifier), or unknown (to be investigated by an operator). This process allows to achieve **high explainability** of the detection results. It has also been proposed in previous work, in the context of network intrusion detection [33].

The operator may inspect uncategorized anomalies to determine if they are false positives or new threats to be added for updating the attack classifier. In both cases, the detectors may be updated.

We point out that the attack classifier may also directly observe real-time data, to detect threats that would be missed using anomaly detection. Such threats may be related, for instance, to security misconfigurations that are common in real-time traffic and hence look as normal (e.g., outdated software, unsafe protocols, etc.). This allows to further **enhance detection accuracy**.

To fight the possible presence of attack patterns in training data used by the anomaly detector (one-class legitimate), there may be a learning filter, that filters real-time data to protect against poisoning attacks.

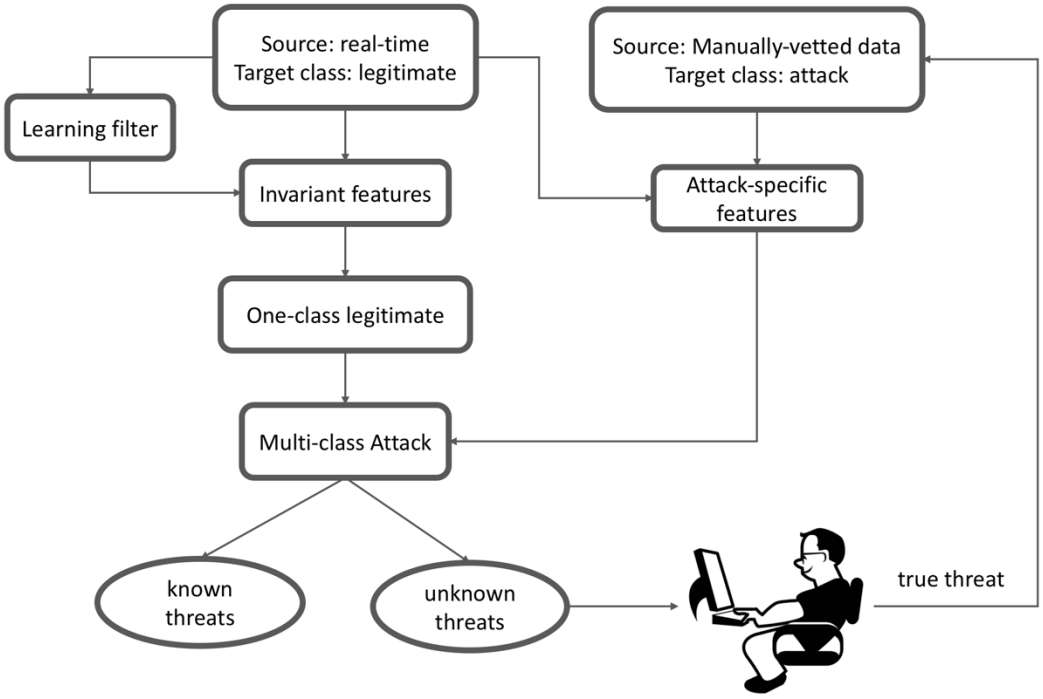

Figure 2. Example of an adversary-aware architecture for threat detection based on Machine Learning

In Section 4.2 we provide a concrete/realistic implementation of this scheme in the context of web security.

4. Real-world case study on web security

To illustrate how the framework presented in Section 3 can be applied, we will refer to a simple, but *real-world* case study on web security. The task is to detect attacks on a specific URI parameter **page**, used by a web application to paginate the results. In Table 1, we show an example of a legitimate input as well as OWASP TOP 10 attacks (2017)[11]: Injection (rank: 1) and Cross-site Scripting (rank: 7). Both attacks must leverage missing or insufficient *input validation* by the underlying web application in order to be successful.

Instance	**Class**
page=34	Legitimate
page=5 order by 1/*	Injection (SQL)
page=<script src="malware.js"/>	Cross-Site Scripting

Table 1. Examples of both legitimate and attack instances on attribute **page**.

4.1 Data

The ML system may acquire data about legitimate and/or attack instances to learn the classifier. In the following, we enumerate some real-world sources of data that can be employed to learn from a target class. For each source, we qualitatively estimate the related properties.

Data choice #	**1**	
Description	Collection of real-world exploits (e.g., from exploit-db.com)	
Target class	Attack	
Accuracy	Low/Medium	The system learned on known attacks may miss attack variants, as well as never-before-seen ones (zero-days).
Robustness	High	Data is validated by a security expert it is highly unlikely that it contains adversarial noise.
Adaptiveness	Low	Manual validation may be necessary.
Explainability	High	Data may be associated to a known exploit, with accurate description of goals, and exploited vulnerabilities (e.g., CVE).

Data choice #	**2**	
Description	HTTP(S) requests. We produce a pattern for each request using the page parameter.	
Target class	Legitimate	
Accuracy	Medium/High	The system learned on normal (legitimate) traffic may accurately detect known attacks as well as never-before-seen ones (zero-days). However, the training data is likely to contain attacks in the wild (noise).
Robustness	Low	The attacker may generate an arbitrary number of attack requests and thus attack patterns
Adaptiveness	High	No need for manual intervention.
Explainability	Low/Medium	Each detection may be associated to an HTTP(S) request, but not to known attacks.

Data choice #	**3**
Description	(public) source IP addresses. We produce a pattern for each IP address using the page parameter.
Target class	Legitimate

[11] https://www.owasp.org/index.php/Category:OWASP Top Ten 2017 Project

Accuracy	Medium/High	The system learned on normal (legitimate) traffic may accurately detect known attacks as well as never-before-seen ones (zero-days). However, the training data is likely to contain attacks in the wild (noise).
Robustness	Medium	It can be considered as more robust than data choice #2. The number of adversarial attack patterns depends on the number of different source IP addresses that the attacker can acquire.
Adaptiveness	High	No need for manual intervention.
Explainability	Low/Medium	Each detection may be associated to an IP address, but not to known attacks.

Data choice #	**4**	
Description	Authenticated user	
Target class	Legitimate	
Accuracy	Medium	The system learned on normal (legitimate) traffic may accurately detect known attacks as well as never-before-seen ones (zero-days). However, the authenticated user may not be sufficiently representative of all legitimate traffic.
Robustness	Medium/High	It can be considered as more robust than data choice #3. The number of adversarial attack patterns depends on the number of authentication credentials the attacker is able to acquire.
Adaptiveness	High	No need for manual intervention.
Explainability	Low/Medium	Each detection may be associated to an authenticated user, but not to known attacks.

Data choice #	**5**	
Data	Trusted authenticated users	
Target class	Legitimate	
Accuracy	Medium	The system learned on normal (legitimate) traffic may accurately detect known attacks as well as never-before-seen ones (zero-days). However, the trusted authenticated user may not be sufficiently representative of all legitimate traffic.
Robustness	High	It can be considered as more robust than data choice #4. The attacker should authenticate him/herself as a trusted user in order to inject noise.
Adaptiveness	Medium/High	No need for manual intervention, except for the definition of trusted users.
Explainability	Low/Medium	Each detection may be associated to a trusted authenticated user, but not to known attacks.

4.2 Feature

The acquired features may or may not exploit information about invariant aspects of attack data. In **Errore. L'origine riferimento non è stata trovata.** we enumerate some real-world features that can be extracted for modelling the inputs of the page parameter. For each source, we qualitatively estimate the related adversarial threat and thus, the associated risk.

In Table 2 we show the output of the feature extraction process for each one of the two choices in **Errore. L'origine riferimento non è stata trovata.**.

Feature choice #	**1**	
Description	Sequence of input characters	
Accuracy	Medium	It may detect malicious/anomalous character sequences, but also expose to classification errors due to legitimate/attack characters never seen in the training phase.
Robustness	Low/Medium	The attacker may introduce attack characters in a poisoning attack or evade detection using attacks with characters never observed in the learning phase.
Adaptiveness	High	No need for manual intervention.
Explainability	Low	The extracted feature does not carry significant information for an operator.

Feature choice #	**2**	
Description	Set of (generalized) characters: integer, alphanumeric words, any other character not generalized	
Accuracy	Medium	It is less exposed to false positives (legitimate characters never seen in the training phase) than feature choice #1. However, it may not detect attacks that simply manipulate the order of characters.
Robustness	Medium	The extracted features embed expert knowledge (invariant aspect): input validation attacks need to use non-alphanumeric (meta)characters, that are thus not generalized.
Adaptiveness	High	No need for manual intervention.
Explainability	Low/Medium	The feature provides some kind of understandable information for an operator.

Feature choice #	**3**	
Description	Input categories/enumeration: e.g., integer, floating point, alphanumeric string, url, path, etc. Each input category describes the allowed alphanumeric characters and well as other constraints, such as range, length etc.	
Accuracy	Medium/High	It may detect advanced attacks that manipulate inputs (e.g., abnormal integer/float range, or path length) that would not be detected using feature #2. At the same time, it may be less prone to classification errors due to characters never seen in the learning phase, with respect to feature #1.
Robustness	Medium	The extracted features embed expert knowledge (invariant aspect): input validation attacks need to use non-alphanumeric (meta)characters, that are thus not generalized.
Adaptiveness	High	No need for manual intervention.
Explainability	Medium	The feature provides understandable information for an operator.

Input Instance	Class	Feature #1	Feature #2	Feature #3
34	Legitimate	34	{INT}	INT
5 order by 1/*	Attack	5SorderSbyS1/*	{INT,W,S,/,*}	W, other chars: {S,/,*}

Table 2. Examples of both legitimate and attack instances on attribute page with the related extracted features: INT=integer, W=alphanumeric word, S=space.

4.3 Classifier

Different classification algorithms may show different degrees of risk. In the following, for each one of the main ML techniques we can employ to implement our detection task, we describe the main threat and risk.

Classification choice #	**1**	
Description	Multi-class attack	
Accuracy	Low/Medium	It may only detect known attacks and a subset of attack variations.
Robustness	Low	The attacker can generate successful exploits (including variants of known attacks) that are considered as outliers.
Adaptiveness	High	No need for manual intervention.
Explainability	High	The classifier may provide indication of the type of the attack as well as of the exploited vulnerabilities (e.g., CVE).

Classification choice #	**2**	
Description	One-class legitimate	
Accuracy	High	It may accurately detect any variation of known attacks or never-before-seen ones.
Robustness	Medium	The attacker should inject a sufficient number of poisoning samples in training data to affect the classifier.
Adaptiveness	High	No need for manual intervention.
Explainability	Medium	The classifier may provide indication of the type of anomaly, but not of the specific attack.

Classification choice #	**3**	
Description	Two-class	
Accuracy	Low/Medium	It may accurately detect any variation of known attacks or never-before-seen ones.
Robustness	Low	The attacker may generate attack samples in regions of the feature space with little training data support (blind spots)
Adaptiveness	High	No need for manual intervention.
Explainability	Low	The classifier may not provide an indication of anomalies/attack details.

4.3.1 Adversary-aware combination of design choices

The design choices outlined in Sections 4.1, 4.2 and 4.3 can be intelligently combined according to the general model proposed in Section 3.1.4. Figure 3 shows a possible solution.

The anomaly detector (one-class legitimate classifier) is learned by processing real-time inputs for the *page* attribute within live HTTP(S) requests associated to trusted users (learning filter, data #3). In this way, the anomaly detector may be able to learn that legitimate inputs for page are of *integer* (INT) type, maybe providing also an expected range for legitimate inputs according to observed traffic (e.g., min: 1, max: 100). This process is reasonably protected against poisoning, because the attacker would need to acquire a sufficiently large number of trusted user credentials to inject poisoning noise, depending on the outlier detection threshold set for the anomaly detector.
The anomaly detector may thus detect any type of input validation attack (i.e., any attack submitting an input different from an integer). A multi-class attack detector may further process the raised anomalies to classify them as known attacks, according the sequence of input characters. If no match is found, the raised

anomalies may be further investigated and categorized by a security operator, to understand if they are false alarms or new kind of input validation threats.

Please note that **fully explainable information** is provided to the operator, both related to legitimate traffic (i.e., the attribute page *should* receive integer inputs) and attacks (i.e., the guessed type of attack: SQL injection, XSS, etc.).

It is also interesting to observe that the attack classifier may also operate directly on the live HTTP(S) request to detect misconfigurations that would not be detected as anomalies. In this case, these may be due to a *badly implemented* web application. For instance, the web application may routinely have an input like this: page=INT. In this case, the anomaly detector may learn that HTML inputs are *normal*. However, an attacker can easily modify the input to inject malicious code (as in a XSS attack): page=<script src="http://evil.com/ex.js">INT</script>. The only way to detect such threats is using applying an attack classifier to page inputs (not anomalies). In this case, a XSS classifier may raise an alert for the presence of HTML code as input which are indeed a *bad coding practice*.

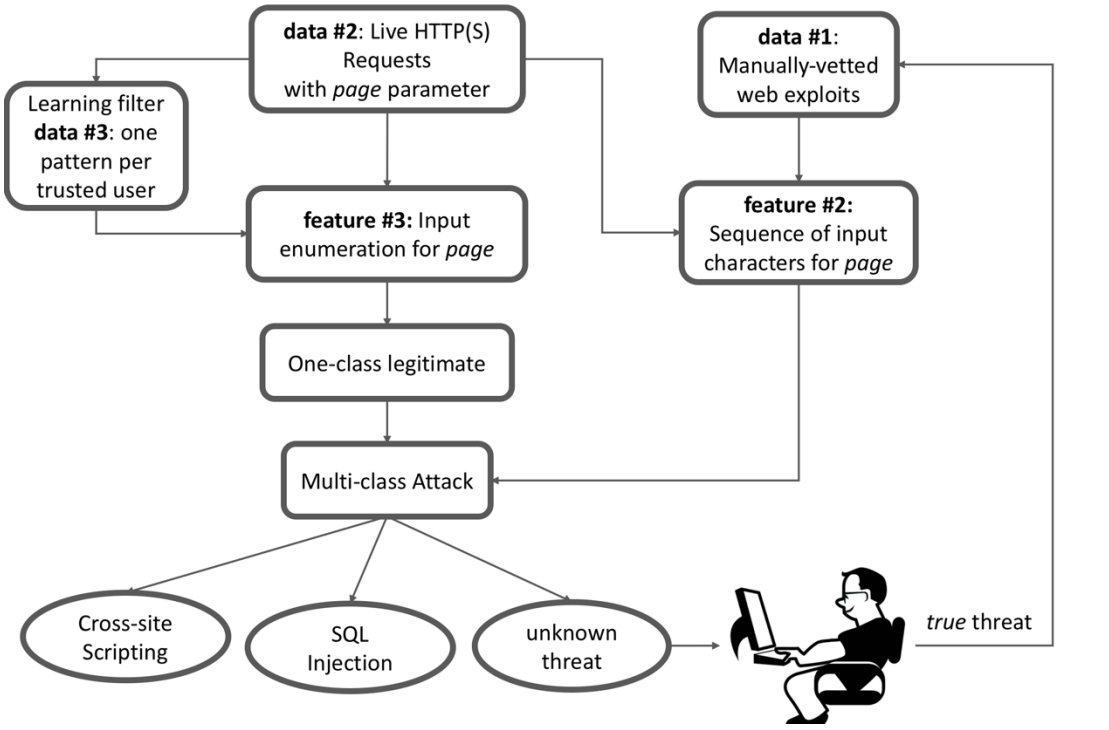

Figure 3. Adversary-aware ML architecture for the case study on web attack detection

5. Evaluation of SIMARGL tools

In the following, we perform an evaluation of threats posed to each product made available by technological partners, according to the framework described in namely, Attack Prophecy (Section 5.1), AI Safe DNS (Section 5.2), Orion Malware (Section 5.3), Cybels Sensor (Section 5.4), IP_ADS (Section 5.5), Red Alert (Section 5.6), Punch Platform (Section 5.7).

5.1 Attack Prophecy

Attack Prophecy natively implements the general ML framework described in Section 3.1.4. Hence, it can natively provide secure, fair and explainable results, according to the guidelines of this deliverable. In particular, multiple anomaly detectors (one-class legitimate classifiers) are learned from legitimate traffic: each anomaly detector focuses on different macro-categories of attacks, on different aspects/fields of HTTP(S) requests. Such macro-categories may include: input validation attacks (injection, cross-site scripting, XML External Entities, HTTP response splitting, etc.), anomalous linked sites (e.g., phishing, malvertising, malicious search engine optimization, etc.), anomalous request rate and server response (bruteforce, probing attacks, and unauthorized content dump), anomalous user authentication (user impersonation, credentials theft, etc.). This allows for high explainability of the detected anomalies. Multiple, specialized anomaly detectors allow one also to focus on intrinsic features related to different kind of attacks. A clear example in the context of input validation attacks has been given in Section 4.

For each anomaly or in general from the raw traffic, Attack Prophecy may also use well-crafted signatures to categorize/classify anomalies or detect threats that are common in the traffic and thus may not be detected using an anomaly-based approach (e.g., server and application misconfigurations). This allows for further improve alert descriptions (explainability) as well as accuracy. Anomalies and alerts may be also automatically aggregated according to a number of features such as the source IP address, ASN, geolocation, type, time, to facilitate the analysis task by user operators. Finally, alerts may undergo automatic verification mechanisms to assess their actual impact/risk on the targeted services (e.g., by analyzing server/app response, version, etc.). These mechanisms are also capable to naturally address overstimulation attacks.

5.2 AI Safe DNS

AI Safe DNS natively implements the general ML framework described in Section 3.1.4. Hence, it can natively provide secure, fair and explainable results, according to the guidelines of this deliverable. In particular, multiple anomaly detectors (one-class legitimate classifiers) are learned from legitimate traffic: each anomaly detector focuses on different macro-categories of attacks, on different aspects/fields of DNS traffic. This allows for high explainability of the detected anomalies. Multiple, specialized anomaly detectors allow one also to focus on intrinsic features related to different kind of attacks.

Sophisticated anomaly-based models are learned from traffic to detect fast-flux and domain-flux botnets, typosquatting attacks and protocol anomalies (that may also indicate covert channels). For each anomaly or in general from the raw traffic, AI Safe DNS may also use well-crafted signatures to categorize/classify anomalies or detect threats that are common in the traffic and thus may not be detected using an anomaly-based approach (e.g., DNS misconfigurations). This allows for further improve alert descriptions (explainability) as well as accuracy. In particular, attack signatures may be related to blacklisted domain names together with the associated malware family or target (e.g., typosquatting against a specific organization), or detection patterns related to protocol misconfiguration and known tools for covert channel creation such as OzymanDNS[12], Iodine[13], etc.

[12] https://dankaminsky.com/2004/07/29/51/

[13] https://code.kryo.se/iodine/

5.3 Orion Malware

Orion malware has been extensively described in deliverable D2.3. It is a comprehensive, sophisticated solution for the detection of malware files. Figure 4 shows the main components of Orion malware according to the proposed secure, fair and explainable ML framework. Orion malware employs a multi-class attack classifier composed by heuristics and signatures based on both static and dynamic features of files. Such attack classifiers have been tested on known malware files (manually-vetted data). The system accepts any file as input (source: *realtime*) and provides a detailed analysis report. Such report may highlight if the file matches with heuristics and signatures associated to known malware, and provide detailed information of the static/dynamic features extracted for the sample. Thus, the Orion malware output is highly **explainable**.

As described in Section 3.1.3, attack classifiers that operate from raw input (as in this case) may be subject to evasion by attack variations and zero-day attacks. In this case, in particular, an attacker may have different ways to evade detection, for instance, using:

- files currently not supported by the analysis engine;
- file manipulation to evade file parsing (static analysis);
- sandbox detection mechanisms to evade dynamic analysis;
- file manipulation to evade detection heuristics/signatures.

Hence, Orion malware is mainly subject to *evasion* threats. As it can be seen from Figure 4, there is no component dedicated to anomaly detection, that may allow one to detect unknown (zero-day) (stego)malware files. To cope with this threat, anomaly detection modules may be built to detect:

- malformed file formats;
- unexpected content (e.g., too much binary data in graphic files);
- parsing errors;
- anomalous external connections/domain resolutions (e.g., IP geolocation requests made by evasive malware to detect sandboxes);
- anomalous dynamic behaviour.

Such detection modules may be implemented in the SIMARGL platform, using the output of Orion malware for a given (legitimate) file and exploiting ML algorithms according to the general ML framework described in Section 3.1.4.

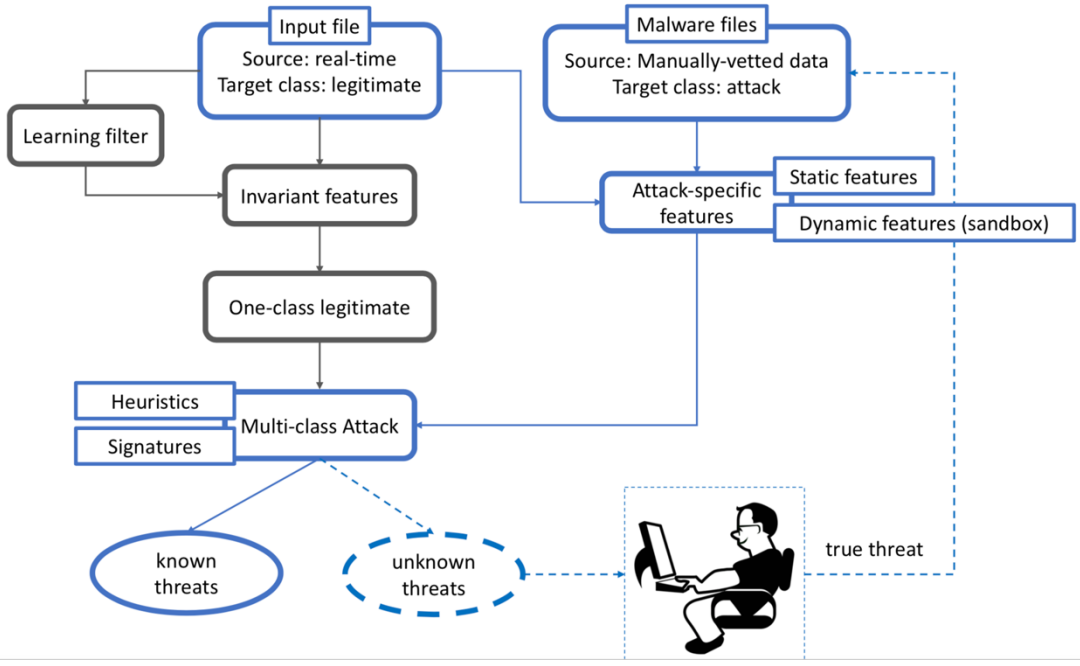

Figure 4. Components of the ML framework for Orion Malware

5.4 Cybels sensor

Cybels sensor has been described in deliverable D2.3. It is a network probe for the detection of network attacks and malicious files. It combines signature-based detection on network traffic and files exchanged over the network with metadata generation to help operators to identify malicious traffic and have a better understanding of the overall traffic semantic. Figure 5 shows the main components of Cybels sensor according to the proposed secure, fair and explainable ML framework. Cybels sensor employs a multi-class attack classifier composed by signatures based on static features observed at the network node. Such attack classifiers have been tested on known network attacks (manually-vetted data). The system passively collects network traffic as input (source: *realtime*) and raises an alert whenever network traffic matches known attacks (signatures). It can also be used to collect high-level features of network traffic (metadata generation). Thus, the Cybels sensor output is **explainable**.

As described in Section 3.1.3, attack classifiers that operate from raw input (as in this case) may be subject to evasion by attack variations and zero-day attacks. In this case, in particular, an attacker may have different ways to evade detection, for instance, by using:

- network attacks/malicious files not supported by current signatures;
- network attacks/malicious files whose relevant features are encrypted, e.g., a malicious file transferred using HTTPS from a compromised web site;
- traffic reconstruction errors (desynchronization, segmentation, encoding variations, etc.) that may also affect file identification/analysis.

Hence, Cybels sensor is mainly subject to *evasion* threats. As it can be seen from Figure 5 there is no component dedicated to anomaly detection, that may allow one to detect unknown (zero-day) network attacks, including covert channels used by (stego)malware. To cope with this threat, anomaly detection modules may be built to detect anomalous traffic (normal profile to be built at the customer premises).

Multiple detection modules may focus on different aspects of the traffic, e.g., source/destination of network traffic, protocol anomalies, amount of traffic towards a destination, time, etc. to detect anomalous network behaviour, including covert channels. Such detection modules may be implemented in the SIMARGL platform, using the output of Cybels sensor for a given normal traffic and exploiting ML algorithms according to the general ML framework described in Section 3.1.4.

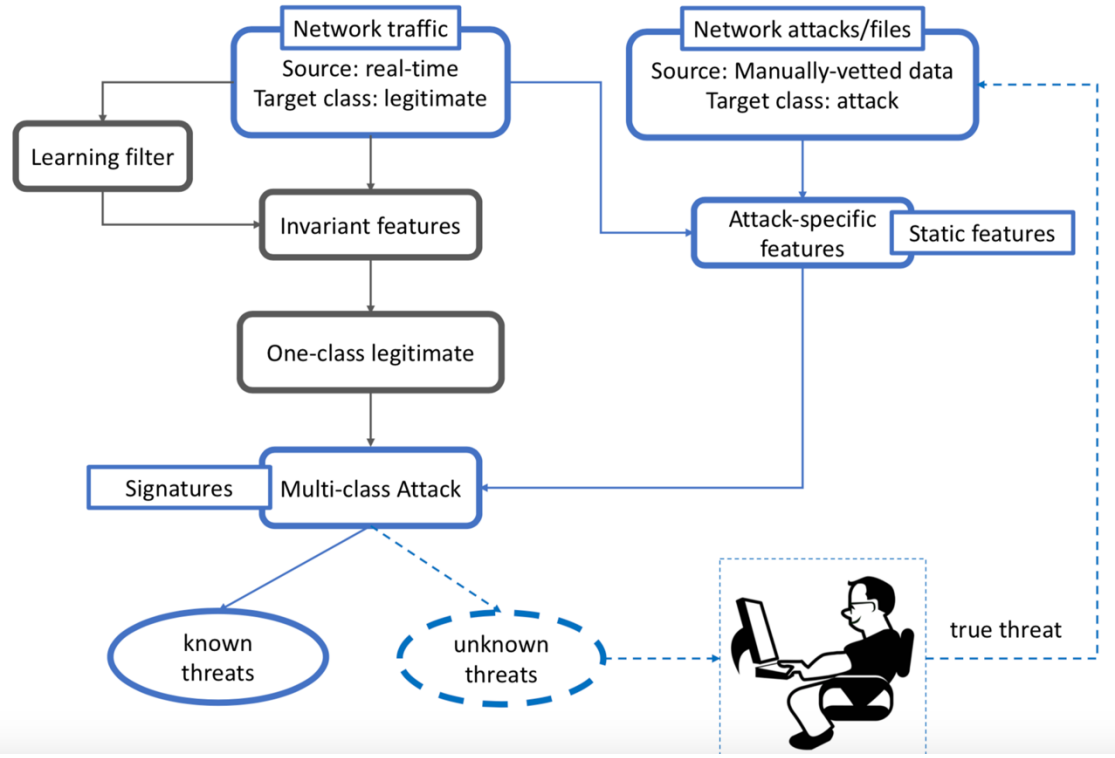

Figure 5. Components of the ML framework for Cybels sensor

5.5 IP_ADS

IP_ADS is a research tool focused on the detection of botnet traffic in computer networks. The data processed by the IP_ADS has mainly a Netflow format[14]. Using Netflow data, the tool extracts additional time-based statistics (features) describing source behavior and/or an entire network connection. Then, a supervised classifier is learned using samples associated to both legitimate and botnet traffic. The system may be able to prefigure the kind of botnet/malware family associated to Netflow traffic, and each feature has been manually crafted and convey a specific meaning, so it can provide **explainable** results.

Figure 6 shows the main components of IP_ADS according to the proposed secure, fair and explainable ML framework. IP_ADS can be viewed as multi-class attack classifier learned with static features observed in network traffic. Such attack classifiers have been tested on known botnet traffic (manually-vetted data). The system passively collects network traffic as input (source: *realtime*) and raises an alert whenever network traffic matches the learned model of known attacks (botnets/malware traffic).

As described in Section 3.1.3, attack classifiers that operate from raw input (as in this case) may be subject to evasion by attack variations and zero-day attacks. In this case, in particular, an attacker may have different ways to evade detection, for instance:

- modifying the behaviour of the botnet traffic/malware to exploit blind spots in the learned supervised classifier (areas of the feature space with little support of training data);
- network attacks/malicious files whose relevant features are encrypted, e.g., a malicious file transferred using HTTPS from a compromised web site;
- traffic reconstruction errors (desynchronization, segmentation, encoding variations, IP address spoofing, etc.) that may affect the reliability of the extracted features.

[14] https://en.wikipedia.org/wiki/NetFlow

Hence, IP_ADS is mainly subject to *evasion* threats. As it can be seen from Figure 6, there is no component dedicated to anomaly detection, that may allow one to detect unknown (zero-day) network attacks, including covert channels used by (stego)malware. To cope with this threat, anomaly detection modules may be built to detect anomalous traffic (normal profile to be built at the customer premises).

Such modules have been already briefly described in the evaluation of Cybels sensor, and in general can strengthen network data analysis (see Section 5.4). IP_ADS together with such anomaly detection modules may be implemented in the SIMARGL lambda architecture (see deliverable D2.3).

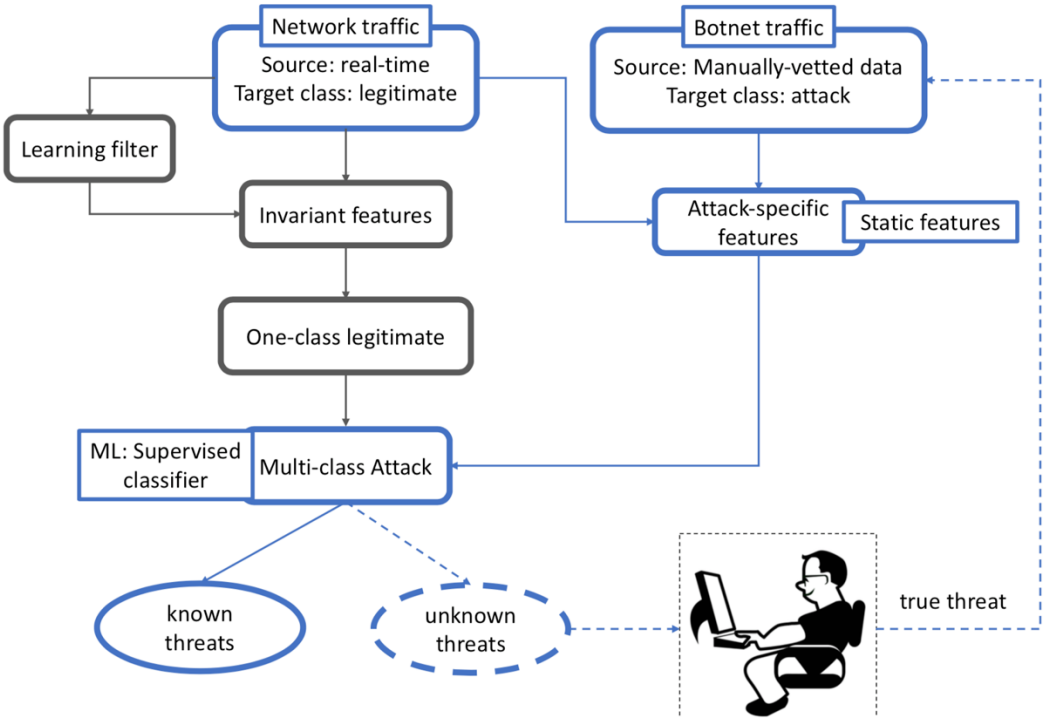

Figure 6. Components of the ML framework for IP_ADS

5.6 Punch Platform

Punch Platform has been extensively described in the deliverable D2.3. In essence, it is a comprehensive, scalable solution for data collection and analytics. Thanks to its flexibility, the platform can be employed to implement the general ML framework described in Section 3.1.4 in the context of the whole SIMARGL platform, where other tools (Attack Prophecy, AI Safe DNS, Cybels sensor, and IP_ADS) are used as input data providers.

5.7 Red Alert

Red Alert (integration platform) can be used to integrate Punch platform with the high-level SIMARGL infrastructure described in deliverable D2.3, Section 2.7. Whereas it is not expected to deliver any detection functionality per se, the integration platform will be extremely important for authentication, access control and visualization aspects. Visualization aspects in particular, will be very important to support **explainability** of the detection results at different abstraction levels, according to the profiles/knowledge of human operators that is expected to work at each level.

5.8 Combination of SIMARGL tools

In this section, we provide an overview of how the tools provided by technical partners in SIMARGL can be integrated into the overall platform. Such integration can exploit the diversity of each tool to greatly extend the detection capability and the reliability of alerts at a higher level (data collection and analytics), according to the high-level SIMARGL infrastructure described in deliverable D2.3, Section 2.7. According to the analysis performed in previous sections, ML algorithms should focus on the implementation anomaly detection modules for network traffic and file analysis. In addition, ML modules may be used to deliver advanced correlation capabilities between the output of different tools, enhancing security (robustness), fairness and explainability of the detection results. The ML framework described in Section 3.1.4. can be used as reference.

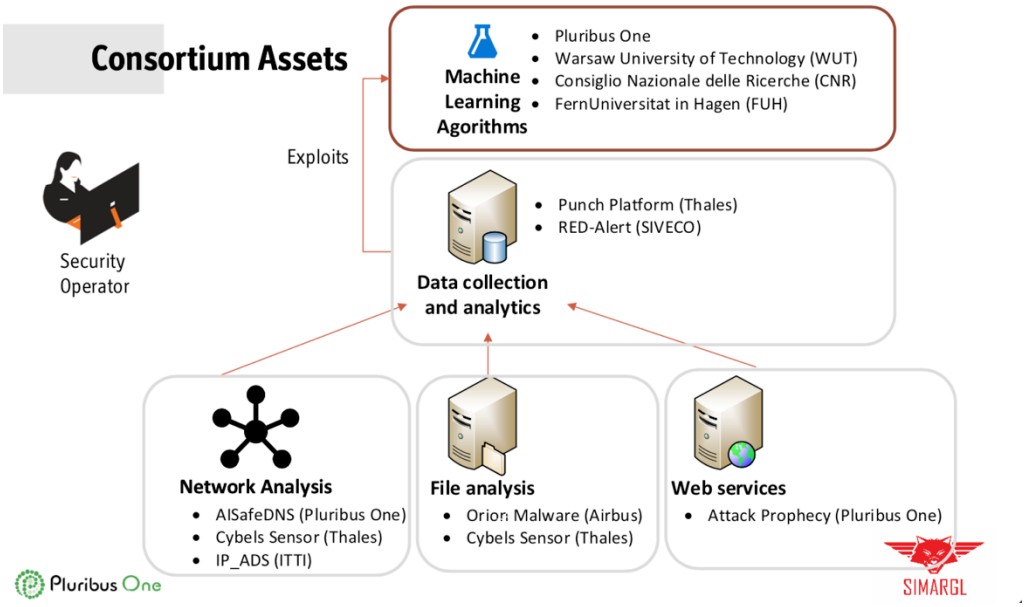

Figure 7. Possible integration of SIMARGL tools

6. Conclusion

In this deliverable, we provided guidelines for the design of *secure, fair and explainable* Machine Learning, together with two main desired properties such as *accuracy* and *adaptability* any ML system should deliver. These guidelines can be exploited by technology providers in the SIMARGL consortium for the improvement their ML tools, their integration, as well as for the development of ML components in the whole SIMARGL platform.

7. References

[1] Zhong Z., A Tutorial on Fairness in Machine Learning, https://towardsdatascience.com/a-tutorial-on-fairness-in-machine-learning-3ff8ba1040cb, last access: 26/11/2019
[2] Del Barrio E., Gamboa, F., Gordaliza P., and Loubes, J. M., Obtaining fairness using optimal transport theory, arXiv preprint arXiv:1806.03195, 2018
[3] Zemel R., Wu Y., Swersky K., Pitassi T., and Dwork, C., Learning fair representations, in International Conference on Machine Learning, pp. 325-333, 2013
[4] Louizos C., Swersky K., Li Y., Welling, M., and Zemel, R., The variational fair autoencoder, arXiv preprint arXiv:1511.00830, 2015
[5] Lum K., and Johndrow, J., A statistical framework for fair predictive algorithms, arXiv preprint arXiv:1610.08077, 2016
[6] Calmon F., Wei D., Vinzamuri B., Ramamurthy K. N., and Varshney K. R., Optimized pre-processing for discrimination prevention, in Advances in Neural Information Processing Systems (pp. 3992-4001), 2017
[7] Woodworth B., Gunasekar S., Ohannessian M. I., and Srebro, N., Learning non-discriminatory predictors, arXiv preprint arXiv:1702.06081, 2017
[8] Zafar M. B., Valera I., Rodriguez M. G., and Gummadi K. P., Fairness constraints: Mechanisms for fair classification, arXiv preprint arXiv:1507.05259, 2015
[9] Zafar M. B., et al., Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment, Proceedings of the 26th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2017.
[10] Agarwal A., Beygelzimer A., Dudík, M., Langford J., and Wallach, H., A reductions approach to fair classification, arXiv preprint arXiv:1803.02453, 2018
[11] Hardt M., Price E., and Srebro N., Equality of opportunity in supervised learning, in Advances in neural information processing systems, 2016
[12] Gall R., Machine Learning Explainability vs Interpretability: Two concepts that could help restore trust in AI, https://www.kdnuggets.com/2018/12/machine-learning-explainability-interpretability-ai.html, last access: 26/11/2019
[13] Došilović F.K., Brčić M., and Hlupić N, Explainable artificial intelligence: A survey, 41st International convention on information and communication technology, electronics and microelectronics (MIPRO). IEEE, 2018
[14] Umang B., et al, Explainable Machine Learning in Deployment, arXiv preprint arXiv:1909.06342, 2019
[15] Albert C., We are ready for Machine Learning Explainability?, https://towardsdatascience.com/we-are-ready-to-ml-explainability-2e7960cb950d, last access: 26/11/2019
[16] Etmann C., et al., On the Connection Between Adversarial Robustness and Saliency Map Interpretability, arXiv preprint arXiv:1905.04172, 2019
[17] Carvalho D. V., Pereira E. M., and Cardoso J. S., Machine Learning Interpretability: A Survey on Methods and Metrics, Electronics 8.8, p. 832, 2019
[18] Molnar C., A guide for making black box models explainable, https://christophm.github.io/interpretable-ml-book, last access: 26/11/2019
[19] Robnik-Šikonja M. and Bohanec M., Perturbation-Based Explanations of Prediction Models, in Human and Machine Learning, Springer, Cham, pp. 159-175, 2018
[20] Silva W., et al., Towards Complementary Explanations Using Deep Neural Networks, in Understanding and Interpreting Machine Learning in Medical Image Computing Applications. Springer, Cham, pp. 133-140, 2018
[21] Honegger M., Shedding Light on Black Box Machine Learning Algorithms: Development of an Axiomatic Framework to Assess the Quality of Methods that Explain Individual Predictions, arXiv, arXiv:1808.05054, 2018
[22] Newsome J., Karp B., and Song D. X., Polygraph: Automatically generating signatures for polymorphic worms, in IEEE Symposium on Security and Privacy. Oakland, CA, USA: IEEE Computer Society, pp. 226–241, 2005

[23] Perdisci R., Dagon D., Lee W., Fogla P., and Sharif M., Misleading worm signature generators using deliberate noise injection, in IEEE Symposium on Security and Privacy. Washington, DC, USA: IEEE Computer Society, 2006.
[24] Newsome J., Karp B., and Song D. X., Paragraph: Thwarting signature learning by training maliciously, in RAID, ser. Lecture Notes in Computer Science, D. Zamboniand C. Krugel, Eds., vol. 4219. Springer, pp. 81–105, 2006
[25] Tong L., Li B., Hajaj C., Xiao C., Zhang N., and Vorobeychik Y., Improving robustness of ML classifiers against realizable evasion attacks using conserved features, in Proceedings of the 28th USENIX Conference on Security Symposium (SEC'19), USENIX Association, Berkeley, CA, USA, pp. 285-302. https://www.usenix.org/conference/usenixsecurity19/presentation/tong, 2019
[26] Pearl J., and Mackenzie D., The Book of Why: The New Science of Cause and Effect *(1st ed.)*, Basic Books, Inc., New York, NY, USA, 2018
[27] Madras D., Creager E., Pitassi T., and Zemel R., Fairness through Causal Awareness: Learning Causal Latent-Variable Models for Biased Data. In Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT* '19), ACM, New York, NY, USA, pp. 349-358, 2019
[28] Smutz C., and Stavrou A., Malicious pdf detection using metadata and structural features, in Proceedings of the 28th Annual Computer Security Applications Conference, ACSAC ’12, 2012
[29] Srndic N., and Laskov P., Detection of malicious pdf files based on hierarchical document structure, in Proceedings of the 20th Annual Network & Distributed System Security Symposium, 2013
[30] Maiorca D., Corona I., and Giacinto G., Looking at the bag is not enough to find the bomb: an evasion of structural methods for malicious PDF files detection, in Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security (ASIA CCS '13), ACM, New York, NY, USA, pp. 119-130, 2013
[31] Corona I., Maiorca D., Ariu D., and Giacinto G., Lux0R: Detection of Malicious PDF-embedded JavaScript code through Discriminant Analysis of API References, in Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop (AISec '14), ACM, New York, NY, USA, pp. 47-57, 2014
[32] Perdisci R., Corona I., Giacinto G., Early Detection of Malicious Flux Networks via Large-Scale Passive DNS Traffic Analysis, IEEE Transactions on Dependable and Secure Computing, vol. 9, issue 5: IEEE Computer Society, Los Alamitos, CA, USA, pp. 714-726, 2012
[33] Bolzoni D., Etalle S., and Hartel P. H., Panacea: Automating attack classification for anomaly-based network intrusion detection systems, in RAID’09: Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection. Berlin, Heidelberg: Springer-Verlag, pp. 1–20, 2009
[34] Barocas S., and Selbst. A. D., Big data's disparate impact, Calif. L. Rev. 104: 671, 2016
[35] Gajane P., and Pechenizkiy M., On formalizing fairness in prediction with machine learning, arXiv preprint arXiv:1710.03184, 2017
[36] Sahil V., and Rubin J., Fairness definitions explained, in IEEE/ACM International Workshop on Software Fairness (FairWare). IEEE, 2018