SoK: Explainable Machine Learning for Computer Security Applications

Azqa Nadeem*, Daniël Vos*, Clinton Cao*, Luca Pajola†, Simon Dieck*, Robert Baumgartner*, Sicco Verwer*
*Delft University of Technology, Email: {azqa.nadeem, d.a.vos, c.s.cao, s.dieck, r.baumgartner-1, s.e.verwer}@tudelft.nl
†University of Padua, Email: {pajola}@math.unipd.it

***Abstract*—Explainable Artificial Intelligence (XAI) is a promising solution to improve the transparency of machine learning (ML) pipelines. We systematize the increasingly growing (but fragmented) microcosm of studies that develop and utilize XAI methods for defensive and offensive cybersecurity tasks. We identify 3 cybersecurity stakeholders, *i.e.,* model users, designers, and adversaries, that utilize XAI for 5 different objectives within an ML pipeline, namely 1) XAI-enabled decision support, 2) applied XAI for security tasks, 3) model verification via XAI, 4) explanation verification & robustness, and 5) offensive use of explanations. We further classify the literature *w.r.t.* the targeted security domain. Our analysis of the literature indicates that many of the XAI applications are designed with little understanding of how they might be integrated into analyst workflows – user studies for explanation evaluation are conducted in only 14% of the cases. The literature also rarely disentangles the role of the various stakeholders. Particularly, the role of the model designer is minimized within the security literature. To this end, we present an illustrative use case accentuating the role of model designers. We demonstrate cases where XAI can help in model verification and cases where it may lead to erroneous conclusions instead. The systematization and use case enable us to challenge several assumptions and present open problems that can help shape the future of XAI within cybersecurity.**

***Index Terms*—explainable artificial intelligence, cybersecurity.**

1. Introduction

Security practitioners are interested in high-performing machine learning (ML) systems that can also explain why they made a decision [1]. However, despite the unprecedented performance achieved by prevailing ML systems, they have been slow to materialize in the security industry. This is because these systems are considered 'black-boxes' due to their lack of transparency — they are notoriously difficult to understand for humans because of their complex configurations and large model sizes. In addition to the lack of understandability, their correctness and robustness can also not be easily verified. For instance, the model might learn incorrect associations (*i.e.,* spurious correlations) from the input data, giving it the illusion of being performant without actually solving the task[1]. The model might also have fatal weaknesses that can be exploited by an adversary to evade detection[2]. For the safety-critical environment of cybersecurity, the usage of such models is less than ideal. In fact, black-box models are not even allowed in regulated fields unless they are supplemented with explanations [6], *e.g.,* courts do not consider model outputs as admissible evidence unless a forensic analyst is able to justify how the output links to the case [7].

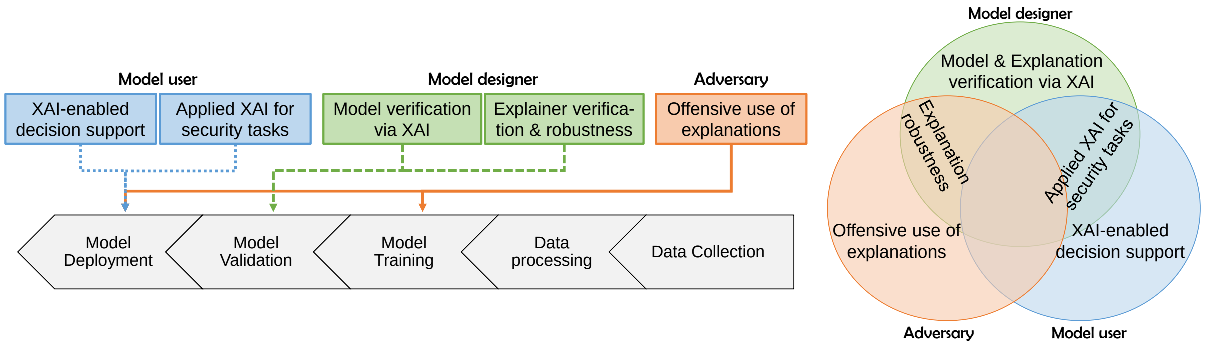
Explainable artificial intelligence (XAI) has been proposed to open the proverbial 'black-box' by making the model internals more human understandable[3]. The first mention of XAI can be traced back to van Lent *et al.* [8] in 2004, while the field really started growing drastically after DARPA announced its XAI program in 2014 [9].

In this paper, we systematize the increasingly growing microcosm of studies that develop and utilize XAI methods for security-specific target domains. We argue that the applications of XAI within cybersecurity are intrinsically different from other domains because cybersecurity works with practical use cases for safety-critical and high-stakes decision-making. The lack of explainability is also a big hurdle for deployment in cybersecurity. In fact, explainability has arguably always been a core tenet in the design of ML pipelines for security [1], [10]. Even the seminal work by van Lent *et al.* uses XAI to explain the behavior of AI-controlled entities in *military simulation games* [8].

In recent years, the security community has actively adopted XAI as a means to increase practitioners' trust [11]. Numerous studies have applied existing XAI methods to security applications [12], [13]. However, recent studies have recognized their shortcomings in addressing the unique pain points of the cybersecurity domain [14]. To this end, several security-specific XAI methods [15]–[17], and evaluation criteria [18], [19] have been developed.

These recent developments have made XAI research within cybersecurity a fast-growing field: while there were merely 42 articles related to *'explainability'*, *'learning'*, and *'cybersecurity'* in 2015, that number has since skyrocketed to 2600+ in 2021, according to Google Scholar. This literature is fragmented across several research communities

1. This is often referred to as the Clever Hans phenomenon [2].
2. Adversarial machine learning studies these cases, *e.g.,* see [3]–[5].
3. The terms 'explainable artificial intelligence', 'explainable ML', and 'interpretable ML' are used interchangeably in the literature.



Model user
XAI-enabled decision support
Applied XAI for security tasks
Model designer
Model verification via XAI
Explainer verification & robustness
Adversary
Offensive use of explanations
Model Deployment
Model Validation
Model Training
Data processing
Data Collection
Model designer
Model & Explanation verification via XAI
Explanation robustness
Applied XAI for security tasks
Offensive use of explanations
XAI-enabled decision support
Adversary
Model user

Figure 1. (Left): The interplay between stakeholder objectives and an ML pipeline. (Right): The interplay between stakeholders and application objectives.

(including ML, security, graphics, and software engineering) with no unified overview. In addition, existing works often use different terminologies to describe XAI, *e.g.,* explicable, accountable, transparent, and understandable.

To the best of our knowledge, this is the first substantial SoK on explainable ML for cybersecurity[4]. By taking a step back, we synthesize insights from the vast body of fragmented literature and identify open areas to stimulate further research in this field.

Following the XAI definitions set forth by Roscher *et al.* [22], we identify three cybersecurity stakeholders, namely *model designers*, *model users*, and *adversaries* that utilize XAI for five distinct objectives within the security literature: (i) XAI-enabled decision support, (ii) applied XAI for security tasks, (iii) model verification via XAI, (iv) explanation verification & robustness, and (v) offensive use of explanations. The interplay between the stakeholders, objectives and the stages of a typical ML pipeline can be seen in Figure 1. Particularly, the stakeholders remain central throughout our discussions. We further categorize the literature *w.r.t* the targeted security domain (*e.g.,* intrusion detection), used ML model, and XAI technique. This taxonomy serves as a guide for finding related literature on XAI for cybersecurity.

After a careful investigation of the available literature, we found that the most popular use of XAI within cybersecurity is to provide decision support – over 48% of the works are classified under *XAI-enabled decision support*. User evaluation is a critical aspect of these studies to ensure that they are sufficiently aligned with existing analyst workflows. However, user studies are conducted in only 14% of the cases, which is alarming since these methods aim to work directly with model users. We propose ideas for mitigating the lack of user studies in §10.

In addition, the stakeholders we identify often have different expertise, and thus require tailored explanations [23]. Yet, the literature we review rarely distinguishes between model users and designers. Note that the role of a model designer in cybersecurity is substantial, *i.e.,* for ensuring the security of the model and the explanation. We tease out the role of model designers in §9: we present a walk-through example of how model designers can utilize XAI to detect and remove spurious features in a network attack detection scenario. The example shows concrete cases where the explanations are helpful, and cases where the model designer should be careful — it became evident that the working knowledge of XAI methods is often crucial in correctly interpreting the explanations. Without such insights, practitioners may draw misleading conclusions.

The rest of the paper is organized as follows: In §2 and §3, we describe the scope and the proposed taxonomy. In §4-§8, we elaborate on the main takeaways from the reviewed literature. In §9, we demonstrate how model designers can utilize XAI to debug their models. In §10, we present open research problems for the future of XAI in cybersecurity.

4. The Explainable Security (XSec) framework proposed by Vigano *et al.* [20] is a non-conventional take on explainability, and does not embed the traditional XAI concepts within the security context. Additionally, although Hariharan *et al.* [21] present a short survey on XAI for cybersecurity, it is limited and covers only a small fraction of the literature.

2. Background and Methodology

Explainable machine learning. ML pipelines either use white-box models, which are inherently *interpretable*, or use black-box models that are explained via *post-hoc explainability*. For instance, linear regression and decision trees are considered white-box, while neural networks and random forest are considered black-box. The output of a post-hoc explainability method is either an *interpretable surrogate model* that approximates the black-box model, or is an explanation of the black-box model in terms of *model components* (*e.g.,* feature importance) or *input examples* (*e.g.,* counterfactuals). Additionally, an explanation can either elucidate how the model prediction is affected by a single data point (*local methods*), or by all data points (*global methods*). Note that interpretable models provide global explanations. Furthermore, most post-hoc methods can be applied to any ML model, making them *model-agnostic*, while interpretable models are sometimes referred to as *model-based explanations*. Figure 2 shows the various terminologies used within the XAI literature.

Method and Scoping. We synthesize available literature that uses explainable ML for (offensive and defensive) cybersecurity tasks. To this aim, we collect relevant literature, construct a taxonomy based on common themes (*i.e.,* application objectives), and classify the literature into those

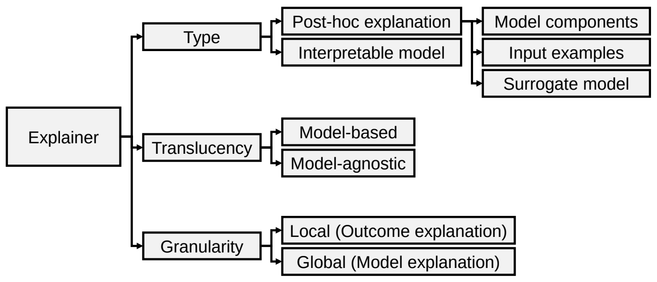


Figure 2. XAI ontology showing the key concepts within XAI.

themes. Applying a reflexive thematic analysis [24], the literature was collected by seven researchers. Each paper was investigated by at least two researchers independently and discussed with all authors during weekly meetings. The code-books were updated as new categories emerged.

We collected peer-reviewed literature that has used explainable models to address cybersecurity problems since 2014, *i.e.,* post-DARPA, by searching popular scientific repositories (*e.g.,* IEEE Xplore) and top security conference proceedings (*e.g.,* Usenix). We used known search terms, *e.g.,* '*explainable*', '*interpretable*', '*artificial intelligence*', '*cybersecurity*', '*robust*', '*offensive*', '*attacks*', and '*trustworthiness*'. To handle the fragmented literature, we expanded our search to include synonyms of the search terms at smaller security and non-security venues. We also included older popular works that try to explain their model without explicitly using XAI *e.g.,* [10], [25], [26]. After carefully reviewing 300+ papers, we select 74 cybersecurity studies to build the taxonomy. Since it is impossible to cover all the available literature in the limited space, we chose representative works from each problem area. As such, there is some overlap with usable security, safety, and robustness literature, but we mainly focus on XAI for security.

3. Systematization

Given a machine learning pipeline from data collection to model deployment, explainable ML can only be applied after the model becomes available. In the literature, XAI is used by model designers and users for objectives such as decision support, and model & explanation verification. In addition, the adversarial threat landscape suggests that an adversary might exploit the explanations for their own gain, or as an additional attack vector. We classify the literature based on the identified stakeholders, application objectives, target domain, model and explainer class. Figure 1 shows an overview of the stakeholder objectives that can be accomplished by applying XAI on a typical ML pipeline.

Stakeholders. We identify three types of stakeholders (or explainees) from the available literature that accomplish distinct objectives by utilizing an XAI technique, even if they use the same ML model. As such, the explanations are tailored to the expertise of the specific stakeholder.

a) Model user is defined as a broad class of personnel who utilize the ML pipeline to improve the defense capacity of an organization, such as an analyst, developer, operator, domain expert, practitioner, or end-user. To this aim, a model user utilizes XAI techniques to better understand the output of a model and make informed decisions, *e.g.,* see [27].

b) Model designer is responsible for engineering the ML pipeline used for a security application. A model designer utilizes XAI techniques to verify that the model works as intended [28], and that the explanations are robust [29].

c) Adversary is a human or an automated agent (malware) that intends to harm an organization by compromising the ML model. An adversary either attacks explanations [30] or utilizes XAI techniques to find better attack vectors [31].

Application objectives. We classify the literature under five objectives based on their use of XAI.

1) XAI-enabled decision support covers techniques that are developed and utilized to support *model users* in making informed decisions with the help of visual analytics dashboards. Human-in-the-loop learning is a core concept in this area where a human analyst interactively guides the learning process [32]. XAI-enabled decision support tools have been utilized for user education [33], false alarm reduction [34], forensic analysis [35], and automated patching [36].

2) Applied XAI for security tasks covers hybrid techniques that are used to better understand a model without necessarily putting a specific stakeholder at the center of the explanation framework. As a rule of thumb, the works that use XAI techniques without explicitly specifying the intended stakeholders fall under this theme. In the available literature, applied XAI has been used for the most diverse set of problems including traffic classification [37] and privacy protection [38]. Game theory has also been utilized for efficient cyber defense resource allocation [39].

3) Model verification via XAI studies techniques that are developed and utilized to help *model designers* debug and validate the correctness of the ML model. For a given black-box model, the proposed techniques either use feature importance/surrogate models to discover spurious features [40], or develop metrics to quantify the model's trustworthiness [41] and interpretability [42].

4) Explanation verification & robustness studies how *model designers* can check the correctness & robustness of post-hoc explanations [43] and interpretable models [44] under natural and adversarial settings. Since interpretable models can be considered as model-based explanations, we also cover their robustness under this theme.

5) Offensive use of explanations studies how *adversaries* can exploit insights provided by XAI techniques for malicious purposes, such as i) compromising the privacy of the model by stealing sensitive data, and ii) compromising the integrity and availability of the model by discovering effective attack vectors. These attacks can be deployed in the model training phase (*e.g.,* poisoning attacks [45]), and model deployment phase (*e.g.,* privacy attacks [46]).

Target domain. We further classify the literature according to the cybersecurity target domain. The literature we reviewed covers the following defensive security domains: *malware detection*, *anomaly detection*, *intrusion detection*, *alert management*, *vulnerability discovery*, *asset prioritiza-*

TABLE 1. Code-book for adopted ML models. 'w' is for white-box models, and 'b' is for black-box models.

Model class	Machine learning algorithms	w	b
CNN	Convolutional neural networks for image data, *e.g.,* ResNet, VGGnet, RPN, and inception network		•
DNN	Deep neural networks for tabular data, *e.g.,* MLP, and auto-encoder		•
GNN	Graph neural networks, *e.g.,* GCN, and graph attention network		•
SeqNN	Sequential neural networks, *e.g.,* RNN, LSTM, and transformers		•
Kernel-SVM	Support vector machine with non-linear kernel		•
Ensemble	Ensemble of models, *e.g.,* random forest, gradient boosting trees, and neural network ensembles		•
LM	Linear models, *e.g.,* logistic (rule) regression, linear rank regression, and linear SVM	•	
RBC	Rule-based classifiers, *e.g.,* decision trees, regular expressions, and BRCG	•	
NB	Naive Bayes and its gaussian variant	•	
Automata	Abstract computing machines, *e.g.,* markov chains, and probabilistic deterministic finite automata	•	
kNN	K-nearest neighbors	•	•
Unsupervised	Clustering algorithms, *e.g.,* HDBSCAN, kmeans, with(out) dimensionality reduction, *e.g.,* self-organizing maps, PCA, t-SNE	•	•

tion, *phishing detection*, *reverse engineering*, *uncertainty estimation*, *traffic classification*, and *privacy protection*. In addition, it covers the following offensive security domains: *adversarial learning*, *evasion attacks*, *privacy attacks*, *poisoning attacks*, and *backdoor injection attacks*. The papers that address generic non-security concepts, such as *adversarial learning*, *anomaly detection*, and *reverse engineering* are further categorized according to the data sources they use, *e.g.,* image, binary, and network traffic.

Model & explainer class. Additionally, we organize the literature according to the broad class of ML models and XAI techniques they use. The models are grouped according to the algorithm and the input data type accepted by the model (*e.g.,* tabular, images). The models are further classified coarsely as either black-box or white-box models, following the consensus of the ML community, see Table 1 for the model code-book. The XAI techniques (called explainers henceforth) are categorized according to their underlying mechanism (*e.g.,* model components, examples, surrogate models), see Table 2 for the explainer code-book.

A summary of the reviewed literature is given in Table 3. The classification in Table 1 reflects the general level of understanding provided by the model class, while Table 3 shows the actual usage of the model. For instance, some studies explicitly treat white-box models as black-box for a model-free approach [12], [41]. Other studies utilize an incomprehensible feature-set (*e.g.,* by representing feature names as integers), turning the pipeline into a black-box [47]. In addition, some works report their methods as being 'interpretable' while utilizing post-hoc explainers for black-box models, *e.g.,* see [16], [48], [49]. Strictly speaking, black-box models cannot be interpretable [22]. Thus, we categorize such works under post-hoc explainability. Note that it is possible to have an interpretable model that also uses a post-hoc explainer, but when a black-box model uses an interpretable model, it is called a surrogate.

TABLE 2. Code-book for adopted explanation methods.

Explainer class	Explanation methods
SHAP	SHAP and its variants, *e.g.,* kernelSHAP
LIME	LIME and its variants, *e.g.,* graphLIME
LEMNA	Non-linear LIME variant for security applications
GNNExplainer	Explanation method for graph neural networks
Grad-based	Gradient based methods, *e.g.,* GradCAM, saliency map, integrated gradients, and layer-wise relevance propagation
Activation	Neuron activations, activation maps and attention
Importance	Feature importance computed using tree-based splitting, feature permutation, and SOM-based dimensionality reduction
Exemplar	Example based explanations, *e.g.,* kNN, prototypes, protoDash
Contrastive	Contrastive explanations, *e.g.,* counterfactuals
Anomaly-score	Custom metric capturing deviation from normalcy, *e.g.,* decoder reconstruction loss
Visual-explanation	Explanation based on visualizing model components or model output for human perception
Sur-RBC	Surrogate rule based classifiers, *e.g.,* decision trees, decision lists, and rule sets
Sur-Mixture	Surrogate mixture linear regression model
Sur-Automata	Surrogate automaton model

4. XAI-enabled Decision Support

Decision support has been a fundamental objective for employing (explainable) ML methods in security workflows. Practitioners have been trying to make their models understandable since before the popularity of XAI [10], [62], [68]. These methods aim to support model users (*e.g.,* security practitioners, developers, end-users) in their decision-making processes. For instance, XAI techniques are popular among Security Operations Center (SOC) analysts to justify the decisions of an ML model in order to comply with their contractual obligations [95]. Other than legal reasons, these techniques are used to reduce practitioner workload by false alarm reduction, automatic patch and threat intelligence generation. XAI also enables human-in-the-loop learning and user education by explaining model outcomes.

Security-specific XAI methods. The specific challenges of cyber data necessitate specialized post-hoc explanation methods for supporting *security analysts* and *developers*. These challenges include concept drift, imbalanced datasets, non-linear decision boundaries, and adversary awareness [96]. For example, LEMNA [15] is a non-linear variant of LIME. It approximates the local (often non-linear) decision boundary using a mixture regression model (*i.e.,* weighted sum of k-linear models) in order to discover the most important features for a classification decision. Giudici *et al.* [6] propose a Shapley-Lorenz decomposition to support ordinal variables that are commonly used for measuring cyber risk. DeepAID [16] is an explanation method for unsupervised settings, such as anomaly detection. It learns a surrogate deterministic finite automaton from any type of deep learning model that allows users to understand the model and improve it, if necessary. Finally, CADE [17] explains why the performance of continual learning systems degrades in operational settings by reporting the features that are most affected by concept drift. It uses the contrastive explanation method: it perturbs features to see which combination affects the distance to the training data

TABLE 3. Summary of XAI literature within cybersecurity. Rows are ordered *w.r.t* Objectives, Target domain, and Year.

			Objectives					Models														Explainers													
Ref.	Year	Target domain	Decision support	Applied XAI	Model verif.	Explanation verif.	Offensive use	Interpretable	Explanation	CNN	DNN	GNN	SeqNN	Kernel-SVM	Ensemble	LM	RBC	NB	Automata	kNN	Unsupervised	SHAP	LIME	LEMNA	GNNExplainer	Grad-based	Activation	Importance	Exemplar	Contrastive	Anomaly-score	Visual-explanation	Sur-RBC	Sur-Mixture	Sur-Automata
[50]	2017	Adv. ML (images) [User education]	•						✓	✓																✓						✓			
[51]	2018	Alert management [FP reduction]	•						✓						✓													✓					✓		
[35]	2021	Alert management [Forensic analysis]	•					✓											✓																
[52]	2022	Alert management [Forensic analysis]	•						✓				✓														✓								
[53]	2022	Alert management [human-in-loop]	•						✓						✓							✓													
[54]	2021	Anomaly det. (network) [FP reduction]	•						✓		✓																				✓				
[34]	2022	Anomaly det. (network) [FP reduction]	•					✓											✓																
[55]	2018	Anomaly det. (sensor) [FP reduction]	•					✓											✓																
[56]	2021	Anomaly det. (sensor) [FP reduction]	•						✓						✓							✓													
[47]	2020	Anomaly det. (syslogs) [FP reduction]	•						✓		✓		✓		✓		✓					✓	✓						✓						
[16]	2021	Anomaly det. [Security XAI]	•	•					✓	✓		✓	✓																						✓
[6]	2021	Asset prioritization [Security XAI]	•	•				✓								✓																			
[57]	2020	Intrusion detection [FP reduction]	•						✓		✓																						✓		
[58]	2020	Intrusion detection [FP reduction]	•						✓		✓											✓													
[13]	2021	Intrusion detection [FP reduction]	•						✓		✓											✓													
[59]	2021	Intrusion detection [FP reduction]	•					✓									✓																		
[60]	2021	Intrusion detection [FP reduction]	•					✓	✓							✓	✓												✓	✓					
[48]	2022	Intrusion detection [Forensic analysis]	•						✓				✓														✓								
[61]	2018	Intrusion detection [Auto-patching]	•					✓	✓		✓					✓														✓					
[33]	2021	Malware analysis [User education]	•						✓												✓											✓			
[10]	2014	Malware detection [User education]	•					✓								✓																			
[62]	2016	Malware detection [User education]	•					✓									✓																		
[63]	2020	Malware detection [User education]	•		•				✓	✓			✓																				✓		
[15]	2018	Malware detection [Security XAI]	•	•					✓		✓		✓																					✓	
[17]	2021	Malware detection [Security XAI]	•	•					✓		✓																			✓					
[64]	2021	Malware detection [User education]	•						✓	✓																✓									
[65]	2021	Malware detection [User education]	•						✓		✓																✓								
[32]	2017	Malware detection [human-in-loop]	•						✓						✓													✓				✓			
[27]	2020	Malware detection [human-in-loop]	•						✓		✓																						✓		
[66]	2021	Phishing detection [User education]	•						✓	✓			✓														✓								
[67]	2021	Phishing detection [User education]	•						✓	✓																						✓			
[68]	2015	Vulnerability disc. [Forensic analysis]	•					✓									✓																		
[40]	2020	Vulnerability disc. [Forensic analysis]	•		•				✓	✓																		✓							
[12]	2020	Vulnerability disc. [Forensic analysis]	•						✓							✓					✓		✓												
[69]	2021	Vulnerability disc. [Forensic analysis]	•						✓			✓													✓										
[36]	2021	Vulnerability disc. [Auto-patching]	•						✓						✓								✓							✓					
[70]	2021	Anomaly det. (sensor) [Feat. analysis]		•					✓												✓							✓							
[71]	2021	Anomaly det. (sensor) [Feat. analysis]		•				✓	✓					✓	✓		✓			✓								✓							
[72]	2018	Anomaly det. (syslogs) [Feat. analysis]		•					✓				✓														✓								
[39]	2021	Asset prioritization [Feat. analysis]		•					✓	✓			✓										✓												
[73]	2020	Intrusion detection [Feat. analysis]		•				✓	✓						✓			✓				✓													
[74]	2019	Malware detection [Feat. analysis]		•	•				✓						✓								✓												
[75]	2020	Malware detection [Feat. analysis]		•	•				✓		✓												✓												
[76]	2021	Malware detection [Feat. analysis]		•				✓	✓		✓			✓	✓		✓			✓			✓					✓							
[77]	2021	Malware detection [Feat. analysis]		•					✓	✓																	✓								
[78]	2022	Malware detection [Feat. analysis]		•	•				✓		✓				✓	✓												✓							
[79]	2017	Phishing detection [Feat. analysis]		•				✓								✓																			
[80]	2021	Privacy protection [Feat. analysis]		•					✓	✓					✓								✓			✓									
[38]	2021	Privacy protection [Feat. analysis]		•					✓						✓							✓													
[28]	2017	Binary analysis [Reverse engineering]		•	•				✓				✓													✓						✓			
[26]	2010	Protocol analysis [Reverse engineering]		•				✓											✓																
[81]	2020	Protocol analysis [Reverse engineering]		•	•			✓											✓																
[37]	2020	Traffic classification [Feat. analysis]		•					✓	✓																		✓							
[82]	2018	Vulnerability disc. [Feat. analysis]		•					✓						✓												✓								
[83]	2019	Vulnerability disc. [Feat. analysis]		•					✓		✓																✓								
[84]	2021	Vulnerability disc. [Feat. analysis]		•	•				✓		✓	✓												✓			✓								
[42]	2022	Malware detection			•			✓	✓		✓			✓	✓			✓		✓													✓		
[41]	2021	Uncertainty estimation			•				✓											✓									✓						
[85]	2018	Adv. ML (images)				•			✓	✓	✓															✓									
[30]	2019	Adv. ML (images)				•	•		✓	✓																✓			✓						
[86]	2019	Adv. ML (images)				•	•		✓	✓																✓									
[87]	2020	Adv. ML (images)				•			✓	✓													✓			✓									
[49]	2020	Adv. ML (images)				•	•		✓	✓																✓									
[88]	2021	Adv. ML (images)				•	•		✓	✓																✓									
[89]	2021	Adv. ML (images)				•	•		✓	✓													✓			✓									
[29]	2018	Adv. ML (images & tabular)				•		✓	✓	✓					✓	✓						✓	✓			✓									
[90]	2020	Adv. ML (pdf)				•	•		✓		✓															✓									
[91]	2021	Backdoor injection					•		✓			✓											✓		✓										
[45]	2021	Backdoor injection					•		✓		✓				✓	✓						✓													
[92]	2019	Evasion attacks					•		✓	✓																✓									
[93]	2021	Evasion attacks					•		✓		✓															✓									
[31]	2021	Privacy & poisoning attacks					•		✓		✓																			✓					
[46]	2021	Privacy attacks					•		✓	✓																✓									
[94]	2021	Privacy attacks					•		✓	✓	✓												✓			✓									

the most, and thus is responsible for causing drift.

Forensic analysis. *Security practitioners* deal with an enormous influx of cyber data that needs to be analyzed for forensic analysis. XAI-enabled triaging techniques have recently been proposed to reduce analyst workload by redirecting their focus on critical events. Wang *et al.* [58] use SHAP to help *security analysts* recognize the unique characteristics of different intrusion attacks. Nadeem *et al.* [35] have developed a novel paradigm of alert-driven attack graphs that show attacker strategies learned from intrusion alerts. *Security analysts* can decide which alerts to investigate by selecting one of the attack graphs. They utilize an interpretable suffix-based automaton model that learns to distinguish between similar but contextually different alerts. van Ede *et al.* [52], on the other hand, use an LSTM to learn this context by capturing the correlation between alerts in an attention vector. Their system clusters attention vectors, capturing the various attack campaigns. *Security analysts* only need to analyze outlier and sampled events from emerging clusters, drastically reducing their workload. Parra *et al.* [48]

also use attention, but from a transformer network in order to perform forensic analysis of syslog entries in a federated learning setup. Finally, Alperin *et al.* [12] propose a focus and context-based visual analytics dashboard to triage vulnerabilities. The dashboard employs topic modeling and LIME to interpret the meaning behind vulnerability scores and further uses t-SNE for clustering similar vulnerabilities. These works support *analysts* in triaging large volumes of cyber data and obtaining actionable intelligence.

Next to triaging, several approaches extract vulnerable patterns from code bases that can serve as actionable intelligence for *software developers* to improve the security of applications. In addition to highlighting the potentially vulnerable source code, Li *et al.* [69] provide a set of data and control dependencies present between code statements, and Zou *et al.* [40] and Yamaguchi *et al.* [68] produce human-understandable rules that can be used for periodic scanning and control. These works are particularly interesting because they address the lack of interpretability in vulnerability discovery methods [97].

False alarm reduction. Although ML systems raise dramatically fewer false alarms than their signature-based counterparts, the false alarm rate is still quite concerning. Analyzing alarms is labor intensive and often leads to 'alert fatigue' — a phenomenon that reduces analyst productivity. XAI methods have been proposed to help analysts better understand alerts and quickly disregard false alarms. To this end, several works have used SHAP to support *network administrators* in understanding the anomalous alerts of intrusion detection systems (IDS) [13], container clouds [47], and industrial control systems [56].

Liu *et al.* [60] propose a self-explainable IDS that uses XAI and an efficient data cleaning pipeline to improve the understandability of alerts by explaining the model internals and its decisions to a *security analyst*. They use both interpretable rule-based models and example-based explanations. Sopan *et al.* [51] propose a dashboard to predict whether an alert is a real threat. The dashboard also provides an explanation of the prediction in the form of an approximated decision path followed by the model and a list of important features. A similar approach is proposed in [57], [59].

A handful of works use anomaly scores to automatically discard anomalous events, thus reducing analyst cognitive load. Akerman *et al.* [98] use image reconstruction loss as an indication of whether artifacts in ADS-B video frames are real threats or false alarms. ADS-B is a protocol used by air traffic controllers to communicate with pilots regarding surrounding objects. By highlighting what might be false alarms, *pilots* can efficiently focus on the mission at hand. Ardito *et al.* [54] use anomaly scores to support *medical staff* in detecting when an attack has occurred on a patient's e-health telemonitoring device. The auto-encoder-based system avoids processing anomalies that can otherwise have devastating effects on a patient's health. Instead, it sends out a validation request to the medical staff.

Finally, automaton-based interpretable models have been developed for *security analysts* to better understand the root cause of alarms. For instance, Lin *et al.* [55] learn an automaton for industrial control systems, while Cao *et al.* [34] learn the same for a Kubernetes cluster. These models can be relatively easily visualized for model-based explanations.

Human-in-the-loop learning. The black-box nature of complex ML pipelines removes human analysts from the learning process entirely. The approaches we discuss here leverage XAI to re-integrate humans in the learning process to further amplify human intelligence [99]. Mahdavifar *et al.* [27] extract a surrogate rule-set from a pre-trained neural network that substitutes the knowledge base of their expert system. *Security analysts* interact with the expert system, which uses the rule-sets to explain classification decisions. These rules are then used to classify unseen security incidents (malware attacks and phishing attempts). Holder *et al.* [100] propose to use ML as a 'virtual analyst' that curates explanations of large datasets by discovering meaningful patterns and presents them to a 'human' *security analyst* so that they can make informed decisions about which data to triage and what actions to take next. Angelini *et al.* [32] propose a visual analytics system for helping *malware analysts* handle 'gray cases' where a model produces a classification with low confidence. The intuition is that the explanations can either enhance the analyst's confidence in the system, if the explanations make sense, or can trigger model improvement, if they do not.

Usability is an important consideration when designing decision support tools for human analysts. As such, visualizations are not always equivalent to explanations — Complex visualizations contribute to cognitive load, which can subvert effective explanations. The design of an explanation system and its ability to integrate into existing analyst workflows are important predictors of its success and usability [53]. The dashboard proposed in [32] is a good example: it explains the reason for malware detection by showing geo-locations of downloaded files and allowing a *malware analyst* to drill deeper into the individual paths of a random forest. However, simultaneously exploring the paths of $\sim$100 decision trees does not make it any easier to decipher what the model is doing. Instead, it is preferable to provide different explanations based on the user's trust, *e.g.,* by providing less explanation when trust is high, and more explanation when trust is low [101].

Takeaway 1: *Visualization is not equivalent to effective explanation. XAI should reduce complexity rather than adding another layer of complex visualizations [53].*

Automated patching. Recent works utilize counterfactual explanations to support *developers* by automatically generating patches for various applications. Wijekoon *et al.* [36] discover vulnerabilities in source code and proactively correct them with the minimal changes necessary. To this end, they use LIME to find the nearest unlike neighbor (*i.e.,* counterfactual explanation) as the most similar code snippet that is not vulnerable and is used as a patch. In another work, Marino *et al.* [61] identify and correct the cause of missed detections and false alarms in IDS. They use adversarial examples that naturally serve as counterfactual explanations, showing the minimal changes required in feature values to correctly classify the (misclassified) security events. They expect that these insights will further improve the performance of their IDS. However, it is unclear how they avoid overfitting since they utilize the knowledge of the test-set to improve their model performance.

User awareness & education. XAI has been utilized to help *model users* understand model outcomes for increased awareness and insecure behavior deterrence.

In the area of malware detection, graphical and natural language explanations have been used to make the model outcome more intuitive. For instance, Nadeem *et al.* [33] build behavioral profiles of malware samples by clustering their network activities. They visualize the overlap in malware profiles as a global dendrogram, which provides insights to *malware analysts* regarding the capabilities of malware samples. Iadarola *et al.* [64] use gradient-based saliency maps to explain the output of their CNN-based malware detector to *malware analysts*. They construct cumulative heatmaps for individual malware families that show visual differences between their disassembled code. Becker *et al.* [63] propose a visual analytics system that enables *malware analysts* to explore how the model views malware samples at different layers by clustering neuron activations.

In terms of textual descriptions, several works use influential features extracted from malware binaries and network traffic to explain why an app is considered malicious to *malware analysts* and *end-users*. To this end, Arp *et al.* [10] use the coefficients of a linear SVM, Wang *et al.* [62] use feature weights from a decision tree, and Wu *et al.* [65] use attention from a multi-layer perceptron.

Close to malware detection, phishing website detection is an important line of work that uses XAI to improve *end-users'* Internet browsing behavior [66], [67]. Phishpedia [67] employs logo detection to generate visual explanations in the form of insightful annotations that warn users when they land on potential phishing websites. Chai *et al.* [66] take one step further by developing a multi-modal learning setup for more accurate phishing website detection. Their attention-based explanations highlight the URL characters, website text, and images that were relevant for the detection.

Finally, in order to educate *security analysts* regarding the impact of adversarial examples on their ML pipelines, Norton *et al.* [50] develop a visualization suite that lets them investigate the impact of gradient-based adversarial attacks.

In summary, XAI-enabled decision support tools assist a wide range of model users in cyber defense, forensic analysis, and threat intelligence generation. Since these tools are intended to integrate within existing workflows, it is necessary to understand their effectiveness via user studies. As such, visual inspection as an evaluation criteria should be avoided since it can often lead to cherry-picking [102].

Takeaway 2: *Qualitative validation is necessary to understand the efficacy of decision support tools. Yet, only 14% of the literature we review performs user studies.*

5. Applied XAI for Security Tasks

The studies classified under applied XAI produce generic explanations for *model users and designers*. These studies typically explain how a model works for a broad range of security-specific target domains. Applied XAI works are different from decision support tools (discussed in §4) in a couple of ways: a) they tend to explain the model, while decision support mainly focuses on model outcomes; b) they rarely specify the intended stakeholders of the explanations; and c) they often implicitly assume that model users and designers have the same expertise. For example, although both Yamaguchi *et al.* [68] and Duan *et al.* [83] use code property graphs for vulnerability discovery, Duan *et al.* only highlight vulnerable code snippets (*applied XAI*) while Yamaguchi *et al.* extract human-understandable rules that improve developers' understanding and enable further interactions with the system (*decision support*).

The works discussed in this section fall under two large groups: influential feature analysis and reverse engineering.

Influential feature analysis. Historically, one of the best ways to achieve good performance was to rank and select the most important hand-crafted features [10], [62]. These works often dissected the model by showing feature importance — a list of features considered influential by the model. Even today, feature importance is one of the most common ways to explain the inner workings of a model. Perhaps, the most common application of this is in malware detection, where feature importance is used to identify the features responsible for misclassifications, *e.g.,* see [74], [76]–[78]. This is particularly important because malware classifiers can be easily fooled by introducing minor, yet carefully selected feature perturbations [103]. Multi-modal learning can be seen as a promising solution for exposure reduction since the multi-view explanations can often sufficiently detect evasion attempts. For instance, Kyadige *et al.* [75] use file path in addition to the Portable Executable (PE) file content, and show that the malware detector can pick up meaningful file path patterns in LIME for accurately identifying malware.

Vulnerability discovery literature also routinely uses feature importance. These methods highlight the lines of code that the model thinks are vulnerable [82]–[84]. Chakraborty *et al.* [84] specifically use LEMNA to highlight vulnerable code since it is meant for security applications.

Many applications for network attack detection utilize feature importance to find impactful features. Existing work has used attention for anomaly detection [72], SHAP for intrusion detection [73], LIME for advanced persistent threat detection [39], partial dependence plots and Self Organizing Map (SOM) based clustering for attacks on cyber physical systems [70], [71]. Li *et al.* [39] also propose a reinforcement learning approach for the efficient allocation of cyber resources. Furthermore, Ahn *et al.* [37] use a unique feature permutation method based on genetic algorithms to identify the most influential features for network traffic classification.

For privacy protection, Chen *et al.* [38] and Gulmezoglu *et al.* [80] use feature importance to analyze the privacy risks of voice assistants and web requests. Chen *et al.* [38] use SHAP to understand the type of fuzzy words in English and Chinese that cause a tree ensemble based wake-up word detector to become falsely triggered. Gulmezoglu *et al.* [80] use LIME and saliency maps to understand the type of web requests that leak side-channel information, enabling website fingerprinting attacks. Website fingerprinting attacks track users by monitoring the unique combination of websites visited by their browsers. Side-channel information, such as performance counters and cache occupancy leaked by the browser can be used to perform this fingerprinting.

For phishing detection, DeltaPhish [79] uses an SVM ensemble. We classify it under post-hoc explainability because although it uses two linear SVMs for textual and image features, the explanations are based on the linear coefficients of just one SVM. This provides insufficient information to interpret what the full model is doing.

Reverse engineering. Reverse engineering is an often overlooked XAI technique. It is commonly used in software engineering to convert black-box systems into white-box alternatives. We, therefore, include a cursory selection of literature to encourage readers to also consider reverse engineering as an explanation method.

There are two popular application scenarios in the available literature: binary analysis and conformance checking. In the first case, a binary executable is considered a black box, and ML is used to recover information lost during the compilation process. For instance, Chua *et al.* [28] develop an RNN-based method to recover function types and signatures from decompiled binary code. They use several post-hoc explainability methods to verify if the model is able to learn concepts that are comparable to an expert's domain knowledge. To this aim, they use t-SNE to visualize semantically similar word embeddings, and saliency maps to understand which instructions are relevant for the recovery of the input functions.

In the second case, a live system generating traces is considered as a black box, and ML is employed to learn an interpretable model from these traces that can be used to derive insights about the black-box system. Discoverer [25] and Prospex [104] are two popular systems for reverse engineering application-level specifications of network protocols. The specifications are often (actively or passively) learned from network traces in the form of state machines. These specifications can be used to understand custom Command and Control (C&C) channels [26] and to discover vulnerabilities in the implementation of popular protocols [81]. Specifically, Fiterau *et al.* [81] use protocol state fuzzing on servers that use the Datagram Transport Layer Security (DTLS) protocol, and discover many functional and non-conformance issues in several implementations.

Takeaway 3: *Although the studies under applied XAI tackle diverse problems, they rarely disentangle model users & designers, which may be problematic if they have varying expertise. Effective explanations are tailored to a specific user [23]. Thus, we encourage the community to spell out their explanation stakeholders.*

6. Model Verification via XAI

When humans interact with AI systems, they start with the assumption that it is near-perfect. Thus, *faith* or *fidelity* is a major constituent of trust in the beginning, which is eventually replaced by reliance and predictability [105]. The reverse is true for the adversarial threat landscape of cybersecurity: reliance and predictability are important constituents of trust since these systems can be attacked. Thus, *model designers* have a critical role in model verification, which is important for building trust with practitioners.

Verifying the correctness of black-box models is notoriously difficult but increasingly necessary for security applications. The abundant literature on adversarial ML has made it apparent that merely relying on performance metrics is a dangerous strategy as the model might have fatal weaknesses that can be exploited by an adversary. Additionally, the model may learn spurious correlations — artifacts unrelated to the security task that allow the learning algorithm to create shortcuts for separating the classes, instead of actually solving the task [92], [106]. These defects make the model *seem performant* without being able to generalize in practice. This is problematic because generalization is a highly desirable property in security as learning-based systems are meant to detect previously unseen threats.

XAI has been shown to reveal spurious correlations [106], and adversarial learning is used for robustifying models. Since robustifying black-box models does not employ XAI, we do not cover it in this paper. Interested readers are well served by Biggio *et al.* [4] and Rosenberg *et al.* [3].

Black-box model debugging. Influential feature analysis and conformance checking are two methods for black-box model verification. Spurious or irrelevant features can be discovered using feature importance methods, *e.g.,* see [40], [84] for vulnerability discovery and [74], [78] for malware detection. In addition, visualizing the internal components of black-box models can also be helpful in identifying sources of bias and misclassifications [63].

Comparing classifier decisions with some notion of ground truth can also be used for model debugging. For instance, Kyadige *et al.* [75] and Chua *et al.* [28] compare model outputs with expert knowledge as a sanity check to ensure that the model works correctly. Fiterau *et al.* [81] compare known protocol specifications with state machines learned from data for conformance checking. In contrast, de Bie *et al.* [41] follow the intuition that samples close to each other have similar predictions. To this end, they compute a trustworthiness metric for regression predictors by comparing the prediction of a given instance with those of its k-nearest neighbors. Their results show that the metric negatively correlates with prediction errors, enabling the *model designer* to distinguish between trustworthy and untrustworthy predictions.

These are rather straightforward methods for identifying spurious features and are generally successful. In fact, spurious feature detection and removal should become more commonplace before deploying new models. Recent work by Arp *et al.* [106] have made a similar recommendation.

We, therefore, draw attention to the critical role of model designers. We provide an illustrative walk-through in §9 and show how model designers can debug their models using commonplace XAI tools.

Surrogate model debugging. Verification of interpretable models is generally easier since they can directly be inspected for defects. Several works, therefore, infer interpretable models from black-box ones. For instance, Dolejš *et al.* [42] learn rule-based surrogate models from black-box models and estimate how similar they both are in terms of the errors they make. Evidently, the mismatch between the two models can be used for fraud, *i.e.,* it is possible to extract fair explanations from known unfair models [107]. In regulated environments where companies are required to supplement their black-box models with explanations, they can be abused to perform *'fairwashing'* — promoting the false perception that a model is fair when it is actually not.

The solution to fairwashing is to learn a certifiably equivalent surrogate model from a given black-box model. For instance, Weiss *et al.* [108] and Koul *et al.* [109] extract equivalent deterministic finite automata from black-box RNNs. These works are somewhat related to the safety verification literature, *e.g.,* [110]–[113], which falls outside our purview. Early evidence also seems to suggest that robust models are more interpretable than their non-robust counterparts [114]. We discuss the robustness of surrogate models in §7 since they serve as model-based explanations.

Takeaway 4: *"When a measure* [e.g., *accuracy] becomes a target, it ceases to be a good measure" [115]. Thus, instead of overly relying on performance metrics, explainability analysis of black-box models should become more commonplace. As such, feature attribution can be used for identifying spurious features [106].*

7. Explanation Verification & Robustness

Like model verification, explanation verification is crucial to ensure that the ML pipeline works correctly. Within the context of cybersecurity, explanation methods form an additional attack vector for *adversaries*. In this section, we show that explanation methods are sensitive to small adversarial perturbations. This includes both post-hoc methods [30], [49], [86], [88]–[90] and interpretable models [116], [117]. The papers specific to the feature importance-based post-hoc explanations identify three problematic traits:

- Classifications and explanations can change tremendously under small perturbations.
- Input samples can be perturbed to cause misclassifications while leaving the explanation unchanged.
- Input samples can be perturbed to change the explanation while leaving the prediction unchanged.

In addition, example-based explanations can also be attacked. Ghorbani *et al.* [30] investigate the effect of adversarial perturbations on exemplars. Exemplars are samples from the training-set whose features most resemble the instance to be explained. The authors find that while keeping the prediction equal, they can cause the top 3 exemplars to be completely different for perturbed samples. This implies that the perturbed samples enter a part of the model with very different latent features.

Dombrowski *et al.* [86] have further exploited the fragility of explanations to perform targeted attacks. They show that by adding imperceptible perturbations to the input image, the generated explanation can be completely controlled by the adversary. These results severely limit the trustworthiness of post-hoc explanation methods.

The fact that models and explainers can be attacked independently opens up a new range of attacks. For instance, malware authors can evade detection while masking the features that they used for evasion. In this case, the generated feature importance maps do not represent the features that are actually important for classification.

Explanation verification is, therefore, an increasingly important line of work for *model designers*. Objective evaluation criteria are required to (a) limit the possibility of cherry-picking via visual inspection [102], (b) ensure that the explanation method works [70], [85], [87], and (c) ensure that the explanations are robust to adversarial attacks [29], [43], [44], [118]–[120]. Warnecke *et al.* [18] and Ganz *et al.* [19] are excellent starting points for evaluation criteria for a wide variety of post-hoc explainers under security settings. Their criteria include descriptive accuracy, sparsity, completeness, stability, efficiency, and robustness.

Explanation debugging. Adebayo *et al.* [85] propose a sanity check for gradient-based explanations: they reset the weights of a neural network to their initial random values and show that some methods still use information from the input, thus revealing when XAI methods trigger on input data patterns rather than on meaningful model behavior. Wickramasinghe *et al.* [70] test the fidelity of attribution methods by perturbing feature values and analyzing their impact on the explanations. Lin *et al.* [87] follow a more direct approach: they test the correctness of different saliency explanations by deliberately injecting artifacts in the input data to see if the explanations detect them. Specifically, they inject backdoor trigger patterns in input images that would naturally result in misclassifications by a CNN. These backdoor triggers serve as ground truth, *i.e.,* the backdoor features are primarily responsible for causing misclassifications, so a robust explanation should identify them. These studies provide interesting ideas for explanation verification.

Post-hoc explanation robustness. Several methods have been proposed for the robustness evaluation of post-hoc explanation methods. For example, Fokkema *et al.* [43] show that robust explanation methods cannot also be recourse sensitive. This means that there will always be model inputs for which the explanations suggest changes that do not end up changing the model's prediction after all[5]. As a solution, the authors suggest using multiple explanations. For example, one might generate multiple counterfactual explanations pointing in different directions.

5. Note that the inputs for which this happens might not occur in practice and that this problem does not exist in linear unbounded models.

In contrast, Alvarez-Melis *et al.* [29] investigate the smoothness of explanations around data points as a measure of robustness. Based on a local version of the Lipschitz constant, they show that the smoothness of model-agnostic explainers, such as LIME and SHAP varies across datasets. They also show that on the MNIST dataset, gradient-based explanations are approximately four times smoother than LIME. This suggests that model-based explanations are more robust than their model-agnostic counterparts. The intuition here is that robust models are smoother and can thus be more easily interpreted by humans [114].

Surrogate model robustness. A handful of works have proposed robust variants of interpretable models, such as linear models and decision trees. For instance, Chen *et al.* [118] use a robust score function to learn decision trees that are robust to adversarial perturbations within a predefined radius. Calzavara *et al.* [119] later define these perturbations using if-then rules and optimize robust decision trees for arbitrary convex loss functions. Recently, Vos *et al.* [44] have further improved the runtime of robust decision tree learning that allows their method to be used in practice.

Logistic regression is considered interpretable as long as it only has a few non-zero weights [121]. Since logistic regression is a restricted case of neural networks, the methods used for robustifying neural networks can be reused. Hayes *et al.* [120] have recently proposed robust and differentially private logistic regression that can be directly optimized.

Takeaway 5: *The correctness and robustness of both, models & explainers must be verified since adversaries can attack them independently. Specifically, explainer evaluation under a security context is very important.*

8. Offensive Use of Explanations

From the offensive security perspective, *adversaries* can also utilize explanation methods to (further) strengthen their attacks. We organize the nefarious uses of explainers through the lens of the classical confidentiality, integrity, and availability (CIA) triad [122]. Adapting the definitions proposed by Papernot *et al.* [5], attacks on *confidentiality* utilize explanations to expose the model structure or the data on which the model was trained. These attacks pose severe privacy risks for sensitive training data, *e.g.,* patient records. Attacks on *integrity* and *availability* utilize explanations to discover knowledge that adversaries can use to induce specific model outcomes of the adversary's choosing and thwart legitimate users from accessing meaningful model outputs.

Confidentiality attacks. Model explanations give the *adversary* additional information about the inner workings of an ML model, making it easier to reconstruct the model and the training data. This is why explanations are seen as privacy vulnerabilities [123].

XAI has been used to strengthen model inversion [46], membership inference [94] and model extraction attacks [31] in the available literature. Model inversion attacks enable adversaries to reconstruct training data from model predictions [5]. Adversaries can achieve even higher performance by also using model explanations. For instance, Zhao *et al.* [46] successfully recover images from the training data using an XAI-aware model inversion attack. They show that feature importance maps generated from layer-wise relevance propagation (LRP) and gradients helped improve image reconstruction and led to increased model inversion performance compared to only using predicted probabilities.

Membership inference attacks assume that an adversary has some inputs and they want to predict whether they were used during training. Shokri *et al.* [94] perform membership inference attacks by using gradient-based explanations. They use the variance of saliency maps as a feature to infer membership and show that it works better than mere random guessing. The performance further improves when using the full explanation instead of just the variance. The paper confirms that gradient-based methods generally perform better than LRP and LIME for privacy attacks.

Finally, model extraction attacks enable adversaries to recover the model's structure and parameters from predictions. Kuppa *et al.* [31] perform model extraction and membership inference attacks by exploiting counterfactual explanations as an additional attack vector. They perform the model extraction attack by learning a surrogate model from known predictions and explanations. In addition, they perform membership inference by comparing the predictions of the target and counterfactual models to infer whether an input belonged to the training data.

Integrity and Availability attacks. Integrity and availability attacks aim to alter the correct functioning of the model. Evasion attacks allow *adversaries* to evade detection. In poisoning attacks, an adversary injects a small percentage of perturbed data to get some desired changes in the learned model. Backdoor attacks are a special kind of poisoning attack where the adversary makes the model sensitive to a pre-specified trigger pattern. Model explanations can be used for gaining knowledge about the features to perturb in order to perform successful adversarial attacks.

Luca *et al.* [92] use integrated gradients to explain the importance assigned to the various fields of binary executables. They use this information to identify a few bytes in the malware header that need to be perturbed in order to successfully evade detection.

Kuppa *et al.* [31] use counterfactual explanations to find the malware features that most heavily impact the classifier decision. They use this knowledge to craft adversarial training samples that efficiently poison the model. Severi *et al.* [45] use SHAP to craft backdoor triggers in malware detectors. Utilizing the explanation, they determine which features to poison, resulting in a success rate of up to three times higher than that of a greedy algorithm that does not use XAI. Xu *et al.* [91] inject backdoors into GNNs by leveraging XAI techniques. They employ GNNExplainer to identify the parts of the graph to attack, and GraphLIME to identify the node features and values to change. Consequently, they achieve a high attack success rate while only decreasing clean sample accuracy by 2.5%.

In two-player competitive games, Wu *et al.* [93] propose a novel adversarial reinforcement learning agent that utilizes

XAI to exploit the weaknesses of its opponent. In such games, the agents take optimal actions according to their policy function, which is often learned using self-play. Using saliency maps, the proposed adversarial agent observes which of their actions the opponent pays the most attention to, and alters them in the next time stamp, thus confusing and manipulating the opponent's actions.

Takeaway 6: *Uniquely attributed to the security domain, adversaries may abuse explanations to strengthen their capabilities. Interventions are needed to stop this abuse. In the meanwhile, model designers should consider limiting access to their explanations.*

9. Use Case: Debugging a Malicious Network Traffic Detector via XAI

In this section, we present an illustrative walk-through of how a model designer might use XAI for model verification. Specifically, we demonstrate how the investigation of influential features and misclassifications can identify problematic or spurious features. The experiments necessitate a sufficient understanding of post-hoc explainers for correct interpretations and highlight the expressive power of interpretable models.

We take the example of a security analyst as the model designer — we assume that the analyst has developed an ML model to detect malicious botnet traffic on their company's network. They have some intuition of how a potential botnet infected device might behave, and thus use XAI to validate whether the model follows that intuition, *e.g.,* by checking whether it uses any spurious features or any strange artifacts from the training data.

Dataset collection. We use the open-source CTU-13 [124] as our experimental dataset. It has 13 scenarios, each containing both benign and malicious Netflow data. The malicious Netflows in each scenario are collected by monitoring virtual machines (VM) infected with real malware. Each Netflow has the following features: start time (StartTime), duration (Dur), protocol (Proto), source port (Sport), Netflow direction (Dir), destination port (Dport), state (State), source type of service (sTos), destination type of service (dTos), total packets (TotPkts), total bytes (TotBytes), and source bytes (SrcBytes). The dataset contains 64,855,215 benign and 1,535,374 malicious Netflows.

Experimental set-up. For the experiments, we have developed a modular XAI pipeline in Python with six models and three explainers. Implementation details are given in appendix. We release the code for reproducibility[6]. For the experiments, we train a gradient boosting machine (GBM) over all the features of the Netflow data, as described in [124]. The GBM achieves a balanced accuracy of 86.4%. We use SHAP, LIME and LEMNA to explain the predictions of the black-box model. We also learn an interpretable decision tree (see Figure 3) to verify whether similar conclusions can be drawn from model-based and

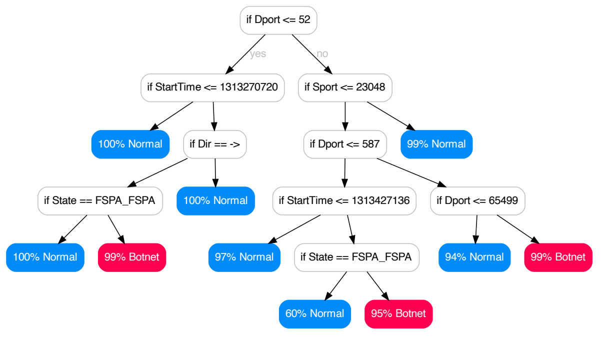


Figure 3. Decision tree for the CTU-13 dataset. It uses StartTime and Sport to differentiate between benign and malicious behavior.

model-agnostic explanations. The decision tree has nine nodes and achieves a balanced accuracy of 83.6%, which is only slightly worse than the GBM. The SHAP summary plot and LEMNA explanations[7] for the GBM are in the appendix. We generate explanations for 140 Netflows from the test-set: 50 true positives (malicious), 50 true negatives (benign), 20 false positives (not malicious), and 20 false negatives (not benign).

XAI for discovering spurious correlations. It is evident from the SHAP summary plot that the GBM exhibits a strong reliance on the destination port, source port, and start time features. The same conclusion can be drawn from the interpretable decision tree in Figure 3.

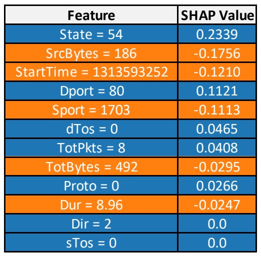
The reliance on the start time and source port features is problematic: start time is problematic because it represents Unix-time, so each new Netflow will have a vastly different feature value compared to the ones seen in the training data, negatively impacting the test accuracy and the model's generalization capabilities. For instance, a benign Netflow that the GBM considers as malicious with a probability of 65% suddenly becomes benign with a 94% probability if we artificially perturb the start time to four weeks earlier. This implies that the model likely learns to predict *when a Netflow is generated*, rather than the Netflow's maliciousness.

Source port is problematic because it typically gets arbitrarily assigned by the operating system, and as such should not be indicative of malicious behavior. However, the CTU-13 dataset uses only a small subset of VM-related port numbers [125], which inadvertently becomes indicative of malicious behavior. This is a common shortcoming of lab-collected datasets [106]. Thus, this feature can also be considered as an artifact of the experimental data. It is noteworthy that start time and source port are perfectly valid features as long as the test-set comes from CTU-13. Since we can not expect real data to follow the same patterns as CTU-13, we consider them as spurious features here.

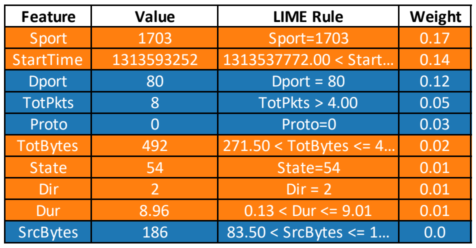
The next logical step is to retrain the model without the spurious features. Doing so lowers the balanced accu-

6. https://github.com/tudelft-cda-lab/xai-pipeline

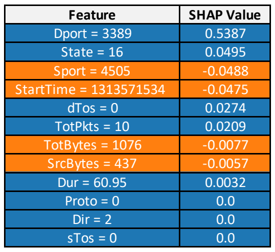
7. We exclude LEMNA from the analysis since the coefficients provide remarkably less insights for class distinction compared to SHAP and LIME.



Feature	SHAP Value
State = 54	0.2339
SrcBytes = 186	-0.1756
StartTime = 1313593252	-0.1210
Dport = 80	0.1121
Sport = 1703	-0.1113
dTos = 0	0.0465
TotPkts = 8	0.0408
TotBytes = 492	-0.0295
Proto = 0	0.0266
Dur = 8.96	-0.0247
Dir = 2	0.0
sTos = 0	0.0



Feature	Value	LIME Rule	Weight
Sport	1703	Sport=1703	0.17
StartTime	1313593252	1313537772.00 < Start...	0.14
Dport	80	Dport = 80	0.12
TotPkts	8	TotPkts > 4.00	0.05
Proto	0	Proto=0	0.03
TotBytes	492	271.50 < TotBytes <= 4...	0.02
State	54	State=54	0.01
Dir	2	Dir = 2	0.01
Dur	8.96	0.13 < Dur <= 9.01	0.01
SrcBytes	186	83.50 < SrcBytes <= 1...	0.0



Feature	SHAP Value
Dport = 3389	0.5387
State = 16	0.0495
Sport = 4505	-0.0488
StartTime = 1313571534	-0.0475
dTos = 0	0.0274
TotPkts = 10	0.0209
TotBytes = 1076	-0.0077
SrcBytes = 437	-0.0057
Dur = 60.95	0.0032
Proto = 0	0.0
Dir = 2	0.0
sTos = 0	0.0

Figure 4. (Left): SHAP explanation for false positive. (Middle): LIME explanation for false positive. (Right): SHAP explanation for false negative. Orange rows contribute positively, and blue rows contribute negatively towards a malicious label.

racy of the GBM and decision tree to 74.4% and 58.1%, respectively. We argue that this is an improvement since the faulty features were making the classifier *appear performant* without being able to generalize to the real world.

Because cyber-data is often noisy, sole reliance on performance metrics is generally meaningless, especially when spurious features are involved. Therefore, we recommend that like ablation studies, identification and removal of spurious features should become a fundamental step in the design of ML pipelines. We showed that influential feature analysis is a rather straightforward way of detecting such spurious features. *Yet, numerous works that have previously used CTU-13 actively include source port and start time as a feature in their pipelines,* e.g., *see [126]–[128].* This is disconcerting because these works are relatively popular with more than 50 citations each. Thus, our experiments call into question such existing works and encourage readers to perform a critical analysis of their ML pipelines before releasing them to the community.

XAI for finding causes of misclassifications. We find that post-hoc explanations must be supplemented with input data statistics to make meaningful inferences regarding the cause of misclassifications. For instance, we analyze a randomly sampled false positive Netflow. The local SHAP explanation (see Figure 4a) shows a heavy reliance on the state value of 54 and source bytes of 186. This information in itself is likely insufficient for an analyst to understand why the model made this mistake. However, combining this information with an analysis of the training data reveals that these feature values appear almost exclusively in malicious samples, thus identifying the cause of the false positive.

In another example, we analyze a randomly sampled false negative Netflow. The local SHAP explanation (see Figure 4c) shows a substantial reliance on the destination port 3389, which is associated with the remote desktop protocol (RDP). Internet-facing RDP servers commonly fall victims to cyber attacks[8], making it a likely indicator of suspicious activity. Yet strangely, the port number 3389 has contributed so heavily towards the opposite. Analyzing the training data reveals that RDP is mostly used by benign hosts in the CTU-13 dataset, due to which the model incorrectly classifies a malicious Netflow as benign. These examples reveal what appears to be sampling and confounding biases in the CTU-13 dataset.

8. http://darktrace.com/blog/botnet-malware-remote-desktop-protocol-rdp-attack

Takeaway 7: *Feature importance explanations do not provide the full picture in isolation. Instead, actionable insights can be obtained by combining the input data together with post-hoc explanations.*

Utility of different explanation types. All explanations are not created equal. Since explanations are meant to explain the behavior of a model, testing the predictability of the model given a few explanations and a new (previously unseen) data instance provides a simple estimation of the explanation's utility. In this sense, there is a clear divide between interpretable models and post-hoc explanations.

For a given interpretable model, such as the decision tree in Figure 3, it is almost trivial to predict how a new instance will be classified by following the decision path. However, the post-hoc explanations are mere approximations, hiding away details about the internal workings of the black-box model. It is difficult to predict how the GBM would classify a new instance, given its LIME and SHAP explanations. For instance, the local SHAP explanations provide feature importance with equality relationships (*e.g.,* see Figure 4a), which makes it impossible to predict how a new instance will be classified, even if it resembles the instances for which explanations are already available. This is because the explanations do not reveal the impact of slight feature perturbations on the classification. We encountered almost the same problem for LIME even though it considers a local neighborhood to prevent this very issue.

Moreover, post-hoc explainers compute their local neighborhoods differently, causing explanations for the same model predictions to differ. Going back to the false positive example, SHAP (Figure 4a) heavily relies on the state feature, while LIME (Figure 4b) assigns a very low importance to it. Also, while SHAP thinks that dTos is important, it does not even appear in LIME. This disagreement problem between feature attribution methods has recently been discovered by Krishna *et al.* [129]. Based on their metric, there is a 25.5% disagreement rate between the top-3 features of SHAP and LIME for our 140 Netflows. This implies that there is a mismatch between the explanations and what the model actually does, and is a stark reminder that explanations should not be interpreted in isolation.

Furthermore, the correct interpretation of post-hoc explanations often relies on how well the explainee understands the underlying mechanisms of the method. For instance, while both local-SHAP and LIME show feature importance, their explanation interpretation can be very different. We found that LIME assigns very low weights to all features for almost all the Netflows. This does not imply that the Netflows similar to the one explained should have the same label, as one would intuitively expect, but rather that LIME is less confident about the prediction given its local surroundings. Thus, an unsuspecting analyst might draw misleading conclusions by overly relying on intuition rather than the understanding of the method [102].

Takeaway 8: *LIME and local-SHAP are both feature attribution methods but their interpretations can be very different. Working knowledge of post-hoc explainers is, thus, cardinal for correctly interpreting the explanations.*

10. Discussion and Open Problems

The conclusions drawn from the literature review and practical use case guide our discussion on open problems, which should stimulate further research in the field of XAI for cybersecurity.

Price of interpretability. From the literature that we have reviewed, merely 24% of the studies adopt interpretable models, while the majority of them focus on applying post-hoc explainability. This may be due to a long-held belief that there exists a trade-off between interpretability and performance, exemplified by the Figure 1 of DARPA's XAI program announcement [9]. However, some works have provided counter-examples showing that interpretable models can achieve performance similar to black-box models [27], [130]. The trade-off between explainability and performance, known as the *'price of interpretability'* [131] is a much-discussed idea in the general ML community, which we believe requires special considerations in cybersecurity. We believe that the presence of an adversary and the prevalence of spurious features will likely make this trade-off less pronounced compared to other fields. Further research is warranted to identify how big the *'price of interpretability'* actually is, and under what circumstances it applies.

Nevertheless, black-box models can be used as a benchmark to guide the search for better interpretable models. Post-hoc explanations can provide actionable intelligence regarding features and parameters for interpretable models. In this way, we view post-hoc explainers and interpretable models as complementary methods rather than alternatives.

The user-study crisis. The lack of qualitative validation among decision support papers is alarming. Since analyst time is expensive, it may be beneficial to develop proxy tasks on which to evaluate new research instead. In the absence of security practitioners, newly developed tools can be peer-reviewed by researchers regarding their usability, *e.g.,* during conference artifact evaluation sessions.

XAI tools for cybersecurity. The uniqueness of cyber data opens up many opportunities for the development of security-specific XAI tools and metrics. For instance, explanation methods are required for unexplored areas, such as temporal data and unsupervised settings. Methods addressing dynamic threat actors, adversarial settings, uncertainty, and heterogeneity also remain unexplored. Moreover, it is unclear how well the security-specific XAI methods compare against existing XAI methods. For instance, recent work has shown that LIME performs better than LEMNA on security applications in terms of descriptive accuracy, sparsity, and efficiency [18]. These kinds of comparisons are critical for establishing the utility of new XAI methods under security settings.

Target domain diversification. The most common target domain for model & explainer evaluation remains malware detection. There are plenty of opportunities to investigate the impact of adversarial examples in other fields, such as cyber operation planning, intrusion and anomaly detection. Secondly, while decision support and applied XAI are popular research themes, there are many open problems in the areas of model & explainer validation. We also foresee opportunities in privacy-preserving XAI methods in order to limit adversaries from abusing explanations.

11. Conclusions

Although the science behind XAI is nascent, security practitioners have been trying to develop human-understandable models since before XAI's popularity. We systematize available research that utilizes explainable models for solving security problems. We identify 3 cybersecurity stakeholders that employ XAI for 5 research objectives within a typical ML pipeline. Distilled from a vast body of diverse literature, this overview streamlines existing research and provides a starting point to reason about open problems.

We found evidence that security literature rarely disentangles model users and designers. In addition, only 24% of the security literature focuses on model & explanation verification. This is problematic because model designers have a critical role in ensuring the correctness and security of an ML pipeline. With regards to model correctness, we specifically provide a walk-through example of how model designers can successfully detect and discard spurious features using SHAP & LIME. At the same time, the example also exposed the disagreement problem between local explanations and showed that SHAP & LIME have different interpretations. Thus, model designers must have a working knowledge of the explanation method in order to draw correct conclusions.

The literature shows how adversaries can target and abuse models & explanations as independent attack vectors, which necessitate specialized evaluation criteria for models & explanations. Moreover, the nefarious uses of explanations can basically compromise all the three principles of the CIA triad, while research on limiting these abuses is non-existent. Finally, the lack of user validation in XAI-enabled decision support shows the substantial margins of improvement within the field of XAI for cybersecurity.

Acknowledgements.. We thank Daniel Meinsma for his contributions to the literature review.

References

[1] Konrad Rieck, Thorsten Holz, Carsten Willems, Patrick Düssel, and Pavel Laskov. Learning and classification of malware behavior. In *DIMVA*, pages 108–125. Springer, 2008.

[2] Laasya Samhita and Hans J Gross. The "clever hans phenomenon" revisited. *Communicative & integrative biology*, 6(6):e27122, 2013.

[3] Ishai Rosenberg, Asaf Shabtai, Yuval Elovici, and Lior Rokach. Adversarial machine learning attacks and defense methods in the cyber security domain. *ACM Computing Surveys (CSUR)*, 54(5):1–36, 2021.

[4] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[5] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P Wellman. Sok: Security and privacy in machine learning. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 399–414. IEEE, 2018.

[6] Paolo Giudici and Emanuela Raffinetti. Explainable ai methods in cyber risk management. *Quality and Reliability Engineering International*, 2021.

[7] Fran Casino, Thomas K. Dasaklis, Georgios P. Spathoulas, Marios Anagnostopoulos, Amrita Ghosal, István Bořočz, Agusti Solanas, Mauro Conti, and Constantinos Patsakis. Research trends, challenges, and emerging topics in digital forensics: A review of reviews. *IEEE Access*, 10:25464–25493, 2022.

[8] Michael Van Lent, William Fisher, and Michael Mancuso. An explainable artificial intelligence system for small-unit tactical behavior. In *Proceedings of the national conference on artificial intelligence*, pages 900–907. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2004.

[9] Gunning D Aha DW. Darpa's explainable artificial intelligence program. *AI Mag*, 40(2):44, 2019.

[10] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. Drebin: Effective and explainable detection of android malware in your pocket. In *Ndss*, volume 14, pages 23–26, 2014.

[11] Sagar Samtani, Murat Kantarcioglu, and Hsinchun Chen. Trailblazing the artificial intelligence for cybersecurity discipline: a multi-disciplinary research roadmap, 2020.

[12] Kenneth B Alperin, Allan B Wollaber, and Steven R Gomez. Improving interpretability for cyber vulnerability assessment using focus and context visualizations. In *IEEE Symposium on Visualization for Cyber Security (VizSec)*, pages 30–39. IEEE, 2020.

[13] Liat Antwarg, Ronnie Mindlin Miller, Bracha Shapira, and Lior Rokach. Explaining anomalies detected by autoencoders using shapley additive explanations. *Expert Systems with Applications*, 186:115736, 2021.

[14] Jose N Paredes, Juan Carlos L Teze, Gerardo I Simari, and Maria Vanina Martinez. On the importance of domain-specific explanations in ai-based cybersecurity systems (technical report). *arXiv preprint arXiv:2108.02006*, 2021.

[15] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, Gang Wang, and Xinyu Xing. Lemna: Explaining deep learning based security applications. In *ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 364–379, New York, NY, USA, 2018. Association for Computing Machinery.

[16] Dongqi Han, Zhiliang Wang, Wenqi Chen, Ying Zhong, Su Wang, Han Zhang, Jiahai Yang, Xingang Shi, and Xia Yin. Deepaid: interpreting and improving deep learning-based anomaly detection in security applications. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3197–3217, 2021.

[17] Limin Yang, Wenbo Guo, Qingying Hao, Arridhana Ciptadi, Ali Ahmadzadeh, Xinyu Xing, and Gang Wang. {CADE}: Detecting and explaining concept drift samples for security applications. In *Proc. of the USENIX Security Symposium*, pages 2327–2344, 2021.

[18] Alexander Warnecke, Daniel Arp, Christian Wressnegger, and Konrad Rieck. Evaluating explanation methods for deep learning in security. pages 158–174, 09 2020.

[19] Tom Ganz, Martin Härterich, Alexander Warnecke, and Konrad Rieck. Explaining graph neural networks for vulnerability discovery. In *ACM Workshop on Artificial Intelligence and Security*, pages 145–156, 2021.

[20] Luca Vigano and Daniele Magazzeni. Explainable security. In *IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 293–300. IEEE, 2020.

[21] Swetha Hariharan, Anusha Velicheti, AS Anagha, Ciza Thomas, and N Balakrishnan. Explainable artificial intelligence in cybersecurity: A brief review. In *International Conference on Security and Privacy*, pages 1–12. IEEE, 2021.

[22] Ribana Roscher, Bastian Bohn, Marco F Duarte, and Jochen Garcke. Explainable machine learning for scientific insights and discoveries. *Ieee Access*, 8:42200–42216, 2020.

[23] Mathias Blumreiter, Joel Greenyer, Francisco Javier Chiyah Garcia, Verena Klös, Maike Schwammberger, Christoph Sommer, Andreas Vogelsang, and Andreas Wortmann. Towards self-explainable cyber-physical systems. In *ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion*, pages 543–548. IEEE, 2019.

[24] Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2):77–101, 2006.

[25] Weidong Cui, Jayanthkumar Kannan, and Helen J Wang. Discoverer: Automatic protocol reverse engineering from network traces. In *Proc. of the USENIX Security Symposium*, pages 1–14, 2007.

[26] Chia Yuan Cho, Domagoj Babi ć, Eui Chul Richard Shin, and Dawn Song. Inference and analysis of formal models of botnet command and control protocols. In *ACM conference on Computer and communications security*, pages 426–439, 2010.

[27] Samaneh Mahdavifar and Ali A Ghorbani. Dennes: deep embedded neural network expert system for detecting cyber attacks. *Neural Computing and Applications*, 32(18):14753–14780, 2020.

[28] Zheng Leong Chua, Shiqi Shen, Prateek Saxena, and Zhenkai Liang. Neural nets can learn function type signatures from binaries. In *Proc. of the USENIX Security Symposium*, pages 99–116, 2017.

[29] David Alvarez-Melis and Tommi S Jaakkola. On the robustness of interpretability methods. *arXiv preprint arXiv:1806.08049*, 2018.

[30] Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. In *AAAI conference on artificial intelligence*, volume 33, pages 3681–3688, 2019.

[31] Aditya Kuppa and Nhien-An Le-Khac. Adversarial xai methods in cybersecurity. *IEEE Transactions on Information Forensics and Security*, 16:4924–4938, 2021.

[32] Marco Angelini, Leonardo Aniello, Simone Lenti, Giuseppe Santucci, and Daniele Ucci. The goods, the bads and the uglies: Supporting decisions in malware detection through visual analytics. In *IEEE Symposium on Visualization for Cyber Security (VizSec)*, pages 1–8. IEEE, 2017.

[33] Azqa Nadeem, Christian Hammerschmidt, Carlos H Gañán, and Sicco Verwer. Beyond labeling: Using clustering to build network behavioral profiles of malware families. In *Malware Analysis Using Artificial Intelligence and Deep Learning*, pages 381–409. Springer, 2021.

[34] Clinton Cao, Agathe Blaise, Sicco Verwer, and Filippo Rebecchi. Learning state machines to monitor and detect anomalies on a kubernetes cluster. In *International Conference on Availability, Reliability and Security*, 2022.

[35] Azqa Nadeem, Sicco Verwer, Stephen Moskal, and Shanchieh Jay Yang. Alert-driven attack graph generation using s-pdfa. *IEEE Transactions on Dependable and Secure Computing*, 2021.

[36] Anjana Wijekoon and Nirmalie Wiratunga. Reasoning with counterfactual explanations for code vulnerability detection and correction. In *AI-Cybersec@ SGAI*, pages 1–13, 2021.

[37] Seyoung Ahn, Jeehyeong Kim, Soo Young Park, and Sunghyun Cho. Explaining deep learning-based traffic classification using a genetic algorithm. *IEEE Access*, 9:4738–4751, 2020.

[38] Yanjiao Chen, Yijie Bai, Richard Mitev, Kaibo Wang, Ahmad-Reza Sadeghi, and Wenyuan Xu. Fakewake: Understanding and mitigating fake wake-up words of voice assistants. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 1861–1883, New York, NY, USA, 2021. Association for Computing Machinery.

[39] Huiling Li, Jun Wu, Hansong Xu, Gaolei Li, and Mohsen Guizani. Explainable intelligence-driven defense mechanism against advanced persistent threats: A joint edge game and ai approach. *IEEE Transactions on Dependable and Secure Computing*, 19(2):757–775, 2021.

[40] Deqing Zou, Yawei Zhu, Hai Jin, Hengkai Ye, Y Zhu, H Ye, Shouhuai Xu, and Zhen Li. Interpreting deep learning-based vulnerability detector predictions based on heuristic searching. *ACM Transactions on Software Engineering and Methodology*, 30, 08 2020.

[41] Kim de Bie, Ana Lucic, and Hinda Haned. To trust or not to trust a regressor: Estimating and explaining trustworthiness of regression predictions. *arXiv preprint arXiv:2104.06982*, 2021.

[42] Jan Dolejš and Martin Jureček. Interpretability of machine learning-based results of malware detection using a set of rules. In *Cybersecurity for Artificial Intelligence*, pages 107–136. Springer, 2022.

[43] Hidde Fokkema, Rianne de Heide, and Tim van Erven. Attribution-based explanations that provide recourse cannot be robust. *arXiv preprint arXiv:2205.15834*, 2022.

[44] Daniël Vos and Sicco Verwer. Efficient training of robust decision trees against adversarial examples. In *International Conference on Machine Learning*, pages 10586–10595. PMLR, 2021.

[45] Giorgio Severi, Jim Meyer, Scott Coull, and Alina Oprea. Explanation-guided backdoor poisoning attacks against malware classifiers. In *Proc. of the USENIX Security Symposium*, pages 1487–1504, 2021.

[46] Xuejun Zhao, Wencan Zhang, Xiaokui Xiao, and Brian Lim. Exploiting explanations for model inversion attacks. In *IEEE/CVF International Conference on Computer Vision*, pages 682–692, 2021.

[47] Rupesh Raj Karn, Prabhakar Kudva, Hai Huang, Sahil Suneja, and Ibrahim M Elfadel. Cryptomining detection in container clouds using system calls and explainable machine learning. *IEEE Transactions on Parallel and Distributed Systems*, 32(3):674–691, 2020.

[48] Gonzalo De La Torre Parra, Luis Selvera, Joseph Khoury, Hector Irizarry, Elias Bou-Harb, and Paul Rad. Interpretable federated transformer log learning for cloud threat forensics. In *Proceedings of the Network and Distributed Systems Security (NDSS) Symposium*, 2022.

[49] Xinyang Zhang, Ningfei Wang, Hua Shen, Shouling Ji, Xiapu Luo, and Ting Wang. Interpretable deep learning under fire. In *Proc. of the USENIX Security Symposium*, 2020.

[50] Andrew P Norton and Yanjun Qi. Adversarial-playground: A visualization suite showing how adversarial examples fool deep learning. In *IEEE symposium on visualization for cyber security (VizSec)*, pages 1–4. IEEE, 2017.

[51] Awalin Sopan, Matthew Berninger, Murali Mulakaluri, and Raj Katakam. Building a machine learning model for the soc, by the input from the soc, and analyzing it for the soc. In *IEEE Symposium on Visualization for Cyber Security (VizSec)*, pages 1–8. IEEE, 2018.

[52] Thijs van Ede, Hojjat Aghakhani, Noah Spahn, Riccardo Bortolameotti, Marco Cova, Andrea Continella, Maarten van Steen, Andreas Peter, Christopher Kruegel, and Giovanni Vigna. Deepcase: Semi-supervised contextual analysis of security events. In *IEEE Symposium on Security and Privacy*, may 2022. 43rd IEEE Symposium on Security and Privacy, S and P 2022, S and P 2022 ; Conference date: 22-05-2022 Through 26-05-2022.

[53] Megan Nyre-Yu, Elizabeth Susan Morris, Blake Cameron Moss, Charles Smutz, and Michael Smith. Explainable ai in cybersecurity operations: Lessons learned from xai tool deployment. In *Usable Security and Privacy (USEC) Symposium*, 2022.

[54] Carmelo Ardito, Tommaso Di Noia, Eugenio Di Sciascio, Domenico Lofù, Andrea Pazienza, and Felice Vitulano. An artificial intelligence cyberattack detection system to improve threat reaction in e-health. In *Italian Conference on Cybersecurity*, 2021.

[55] Qin Lin, Sridha Adepu, Sicco Verwer, and Aditya Mathur. Tabor: A graphical model-based approach for anomaly detection in industrial control systems. In *asia conference on computer and communications security*, pages 525–536, 2018.

[56] Chanwoong Hwang and Taejin Lee. E-sfd: Explainable sensor fault detection in the ics anomaly detection system. *IEEE Access*, 9:140470–140486, 2021.

[57] Mateusz Szczepański, Michał Choraś, Marek Pawlicki, and Rafał Kozik. Achieving explainability of intrusion detection system by hybrid oracle-explainer approach. In *IJCNN*, pages 1–8. IEEE, 2020.

[58] Maonan Wang, Kangfeng Zheng, Yanqing Yang, and Xiujuan Wang. An explainable machine learning framework for intrusion detection systems. *IEEE Access*, 8:73127–73141, 2020.

[59] Basim Mahbooba, Mohan Timilsina, Radhya Sahal, and Martin Serrano. Explainable artificial intelligence (xai) to enhance trust management in intrusion detection systems using decision tree model. *Complexity*, 2021, 2021.

[60] Hong Liu, Chen Zhong, Awny Alnusair, and Sheikh Rabiul Islam. Faixid: a framework for enhancing ai explainability of intrusion detection results using data cleaning techniques. *Journal of Network and Systems Management*, 29(4):1–30, 2021.

[61] Daniel L Marino, Chathurika S Wickramasinghe, and Milos Manic. An adversarial approach for explainable ai in intrusion detection systems. In *Annual Conference of the IEEE Industrial Electronics Society*, pages 3237–3243. IEEE, 2018.

[62] Shanshan Wang, Zhenxiang Chen, Lei Zhang, Qiben Yan, Bo Yang, Lizhi Peng, and Zhongtian Jia. Trafficav: An effective and explainable detection of mobile malware behavior using network traffic. In *IEEE/ACM International Symposium on Quality of Service*, pages 1–6. IEEE, 2016.

[63] Franziska Becker, Arthur Drichel, Christoph Müller, and Thomas Ertl. Interpretable visualizations of deep neural networks for domain generation algorithm detection. In *IEEE Symposium on Visualization for Cyber Security (VizSec)*, pages 25–29. IEEE, 2020.

[64] Giacomo Iadarola, Fabio Martinelli, Francesco Mercaldo, and Antonella Santone. Towards an interpretable deep learning model for mobile malware detection and family identification. *Computers & Security*, 105:102198, 2021.

[65] Bozhi Wu, Sen Chen, Cuiyun Gao, Lingling Fan, Yang Liu, Weiping Wen, and Michael R Lyu. Why an android app is classified as malware: Toward malware classification interpretation. *ACM TOSEM*, 30(2):1–29, 2021.

[66] Yidong Chai, Yonghang Zhou, Weifeng Li, and Yuanchun Jiang. An explainable multi-modal hierarchical attention model for developing phishing threat intelligence. *IEEE Transactions on Dependable and Secure Computing*, 19(2):790–803, 2021.

[67] Yun Lin, Ruofan Liu, Dinil Mon Divakaran, Jun Yang Ng, Qing Zhou Chan, Yiwen Lu, Yuxuan Si, Fan Zhang, and Jin Song Dong. Phishpedia: a hybrid deep learning based approach to visually identify phishing webpages. In *Proc. of the USENIX Security Symposium*, pages 3793–3810, 2021.

[68] Fabian Yamaguchi, Alwin Maier, Hugo Gascon, and Konrad Rieck. Automatic inference of search patterns for taint-style vulnerabilities. In *IEEE symposium on security and privacy*, pages 797–812. IEEE, 2015.

[69] Yi Li, Shaohua Wang, and Nguyen Tien. Vulnerability detection with fine-grained interpretations. pages 292–303, 08 2021.

[70] Chathurika S Wickramasinghe, Kasun Amarasinghe, Daniel L Marino, Craig Rieger, and Milos Manic. Explainable unsupervised machine learning for cyber-physical systems. *IEEE Access*, 9:131824–131843, 2021.

[71] Carmelo Ardito, Yashar Deldjoo, Eugenio Di Sciascio, and Fatemeh Nazary. Revisiting security threat on smart grids: accurate and interpretable fault location prediction and type classification. In *Italian Conference on CyberSecurity*, 2021.

[72] Andy Brown, Aaron Tuor, Brian Hutchinson, and Nicole Nichols. Recurrent neural network attention mechanisms for interpretable system log anomaly detection. In *Workshop on Machine Learning for Computing Systems*, pages 1–8, 2018.

[73] Abel A. Reyes, Francisco D. Vaca, Gabriel A. Castro Aguayo, Quamar Niyaz, and Vijay Devabhaktuni. A machine learning based two-stage wi-fi network intrusion detection system. *Electronics*, 9(10), 2020.

[74] Sherin Mary Mathews. Explainable artificial intelligence applications in nlp, biomedical, and malware classification: a literature review. In *Intelligent computing-proceedings of the computing conference*, pages 1269–1292. Springer, 2019.

[75] Adarsh Kyadige, Ethan M. Rudd, and Konstantin Berlin. Learning from context: A multi-view deep learning architecture for malware detection. In *IEEE Security and Privacy Workshops (SPW)*, pages 1–7, 2020.

[76] Nourhène Ben Rabah, Bénédicte Le Grand, and Manuele Kirsch Pinheiro. Iot botnet detection using black-box machine learning models: the trade-off between performance and interpretability. In *IEEE International Conference on Enabling Technologies*, pages 101–106, 2021.

[77] Martin Kinkead, Stuart Millar, Niall McLaughlin, and Philip O'Kane. Towards explainable cnns for android malware detection. *Procedia Computer Science*, 184:959–965, 2021.

[78] Vasilios Koutsokostas, Nikolaos Lykousas, Theodoros Apostolopoulos, Gabriele Orazi, Amrita Ghosal, Fran Casino, Mauro Conti, and Constantinos Patsakis. Invoice #31415 attached: Automated analysis of malicious microsoft office documents. *Computers & Security*, 114:102582, 2022.

[79] Igino Corona, Battista Biggio, Matteo Contini, Luca Piras, Roberto Corda, Mauro Mereu, Guido Mureddu, Davide Ariu, and Fabio Roli. Deltaphish: Detecting phishing webpages in compromised websites. In *European Symposium on Research in Computer Security*, pages 370–388. Springer, 2017.

[80] Berk Gulmezoglu. Xai-based microarchitectural side-channel analysis for website fingerprinting attacks and defenses. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2021.

[81] Paul Fiterau-Brostean, Bengt Jonsson, Robert Merget, Joeri De Ruiter, Konstantinos Sagonas, and Juraj Somorovsky. Analysis of {DTLS} implementations using protocol state fuzzing. In *Proc. of the USENIX Security Symposium*, pages 2523–2540, 2020.

[82] Rebecca Russell, Louis Kim, Lei Hamilton, Tomo Lazovich, Jacob Harer, Onur Ozdemir, Paul Ellingwood, and Marc McConley. Automated vulnerability detection in source code using deep representation learning. pages 757–762, 12 2018.

[83] Xu Duan, Jingzheng Wu, Shouling Ji, Zhiqing Rui, Tianyue Luo, Mutian Yang, and Yanjun Wu. Vulsniper: Focus your attention to shoot fine-grained vulnerabilities. pages 4665–4671, 08 2019.

[84] Saikat Chakraborty, Rahul Krishna, Yangruibo Ding, and Baishakhi Ray. Deep learning based vulnerability detection: Are we there yet. *IEEE Transactions on Software Engineering*, PP:1–1, 06 2021.

[85] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.

[86] Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. Explanations can be manipulated and geometry is to blame. *Advances in Neural Information Processing Systems*, 32, 2019.

[87] Yi-Shan Lin, Wen-Chuan Lee, and Z Berkay Celik. What do you see? evaluation of explainable artificial intelligence (xai) interpretability through neural backdoors. *arXiv preprint arXiv:2009.10639*, 2020.

[88] Antonio Galli, Stefano Marrone, Vincenzo Moscato, and Carlo Sansone. Reliability of explainable artificial intelligence in adversarial perturbation scenarios. In *International Conference on Pattern Recognition*, pages 243–256. Springer, 2021.

[89] Marzyeh Ghassemi, Luke Oakden-Rayner, and Andrew L Beam. The false hope of current approaches to explainable artificial intelligence in health care. *The Lancet Digital Health*, 3(11):e745–e750, 2021.

[90] Aditya Kuppa and Nhien-An Le-Khac. Black box attacks on explainable artificial intelligence (xai) methods in cyber security. In *IJCNN*, pages 1–8. IEEE, 2020.

[91] Jing Xu, Minhui Xue, and Stjepan Picek. Explainability-based backdoor attacks against graph neural networks. In *ACM Workshop on Wireless Security and Machine Learning*, pages 31–36, 2021.

[92] Demetrio Luca, Battista Biggio, Lagorio Giovanni, Fabio Roli, and Armando Alessandro. Explaining vulnerabilities of deep learning to adversarial malware binaries. In *Italian Conference on Cyber Security*, volume 2315, 2019.

[93] Xian Wu, Wenbo Guo, Hua Wei, and Xinyu Xing. Adversarial policy training against deep reinforcement learning. In *Proc. of the USENIX Security Symposium*, pages 1883–1900, 2021.

[94] Reza Shokri, Martin Strobel, and Yair Zick. On the privacy risks of model explanations. In *AAAI/ACM Conference on AI, Ethics, and Society*, pages 231–241, 2021.

[95] Azqa Nadeem, Shanchieh Jay Yang, and Sicco Verwer. Learning about the adversary. 2022.

[96] Azqa Nadeem, Vera Rimmer, Wouter Joosen, and Sicco Verwer. Intelligent malware defenses. In *Security and Artificial Intelligence*, pages 217–253. Springer, 2022.

[97] Triet Le, Huaming Chen, and Muhammad Ali Babar. A survey on data-driven software vulnerability assessment and prioritization. 07 2021.

[98] Sefi Akerman, Edan Habler, and Asaf Shabtai. Vizads-b: Analyzing sequences of ads-b images using explainable convolutional lstm encoder-decoder to detect cyber attacks. *arXiv preprint arXiv:1906.07921*, 2019.

[99] Tien N Nguyen and Raymond Choo. Human-in-the-loop xai-enabled vulnerability detection, investigation, and mitigation. In *IEEE/ACM ASE*, pages 1210–1212. IEEE, 2021.

[100] Eric Holder and Ning Wang. Explainable artificial intelligence (xai) interactively working with humans as a junior cyber analyst. *Human-Intelligent Systems Integration*, pages 1–15, 2021.

[101] Sule Anjomshoae, Amro Najjar, Davide Calvaresi, and Kary Främling. Explainable agents and robots: Results from a systematic literature review. In *AAMAS*, pages 1078–1088. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[102] Matthew L Leavitt and Ari Morcos. Towards falsifiable interpretability research. *arXiv preprint arXiv:2010.12016*, 2020.

[103] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial examples for malware detection. In *European symposium on research in computer security*, pages 62–79. Springer, 2017.

[104] Paolo Milani Comparetti, Gilbert Wondracek, Christopher Kruegel, and Engin Kirda. Prospex: Protocol specification extraction. In *IEEE symposium on security and privacy*, pages 110–125. IEEE, 2009.

[105] Claire Nicodeme. Build confidence and acceptance of ai-based decision support systems-explainable and liable ai. In *International Conference on Human System Interaction*, pages 20–23. IEEE, 2020.

[106] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. Dos and don'ts of machine learning in computer security. In *Proc. of the USENIX Security Symposium*, 2022.

[107] Ulrich Aivodji, Hiromi Arai, Olivier Fortineau, Sébastien Gambs, Satoshi Hara, and Alain Tapp. Fairwashing: the risk of rationalization. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 161–170. PMLR, 09–15 Jun 2019.

[108] Gail Weiss, Yoav Goldberg, and Eran Yahav. Extracting automata from recurrent neural networks using queries and counterexamples. In *International Conference on Machine Learning*, pages 5247–5256. PMLR, 2018.

[109] Anurag Koul, Sam Greydanus, and Alan Fern. Learning finite state representations of recurrent policy networks. *arXiv preprint arXiv:1811.12530*, 2018.

[110] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[111] Weiming Xiang, Patrick Musau, Ayana A Wild, Diego Manzanas Lopez, Nathaniel Hamilton, Xiaodong Yang, Joel Rosenfeld, and Taylor T Johnson. Verification for machine learning, autonomy, and neural networks survey. *arXiv preprint arXiv:1810.01989*, 2018.

[112] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, 37:100270, 2020.

[113] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *IEEE symposium on security and privacy*, pages 3–18. IEEE, 2018.

[114] Andrew Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[115] James Clear. *Atomic habits: An easy & proven way to build good habits & break bad ones*. Penguin, 2018.

[116] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

[117] Daniel Lowd and Christopher Meek. Adversarial learning. In *ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 641–647, 2005.

[118] Hongge Chen, Huan Zhang, Duane Boning, and Cho-Jui Hsieh. Robust decision trees against adversarial examples. In *International Conference on Machine Learning*, pages 1122–1131. PMLR, 2019.

[119] Stefano Calzavara, Claudio Lucchese, Gabriele Tolomei, Seyum Assefa Abebe, and Salvatore Orlando. Treant: training evasion-aware decision trees. *Data Mining and Knowledge Discovery*, 34(5):1390–1420, 2020.

[120] Jamie Hayes, Borja Balle, and M Pawan Kumar. Learning to be adversarially robust and differentially private. *arXiv preprint arXiv:2201.02265*, 2022.

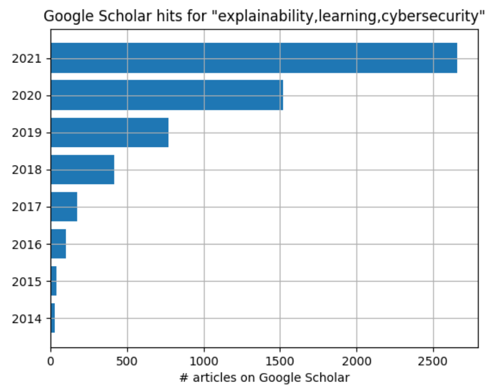


Figure A.1. Prevalence of XAI literature from 2014-2021.

[121] Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[122] Charles P Pfleeger and Shari Lawrence Pfleeger. *Analyzing computer security: A threat/vulnerability/countermeasure approach*. Prentice Hall Professional, 2012.

[123] Patrick Hall, Navdeep Gill, and Nicholas Schmidt. Proposed guidelines for the responsible use of explainable machine learning, 2019.

[124] S. García, M. Grill, J. Stiborek, and A. Zunino. An empirical comparison of botnet detection methods. *Computers & Security*, 45:100–123, 2014.

[125] Clinton Cao, Annibale Panichella, Sicco Verwer, Agathe Blaise, and Filippo Rebecchi. Encoding netflows for state-machine learning. *arXiv preprint arXiv:2207.03890*, 2022.

[126] Miguel Nicolau, James McDermott, et al. Learning neural representations for network anomaly detection. *IEEE transactions on cybernetics*, 49(8):3074–3087, 2018.

[127] Riaz Ullah Khan, Xiaosong Zhang, Rajesh Kumar, Abubakar Sharif, Noorbakhsh Amiri Golilarz, and Mamoun Alazab. An adaptive multi-layer botnet detection technique using machine learning classifiers. *Applied Sciences*, 9(11):2375, 2019.

[128] Rojalina Priyadarshini and Rabindra Kumar Barik. A deep learning based intelligent framework to mitigate ddos attack in fog environment. *Journal of King Saud University-Computer and Information Sciences*, 2019.

[129] Satyapriya Krishna, Tessa Han, Alex Gu, Javin Pombra, Shahin Jabbari, Steven Wu, and Himabindu Lakkaraju. The disagreement problem in explainable machine learning: A practitioner's perspective. *arXiv preprint arXiv:2202.01602*, 2022.

[130] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[131] Dimitris Bertsimas, Arthur Delarue, Patrick Jaillet, and Sebastien Martin. The price of interpretability. *arXiv preprint arXiv:1907.03419*, 2019.

Appendix

XAI pipeline. We develop a modular XAI pipeline in Python (since it has in-built support for many popular models and explainers). The pipeline has three components: (1) The parser parses the input data (train and test) in either CSV or NumPy array format. The user can specify to the

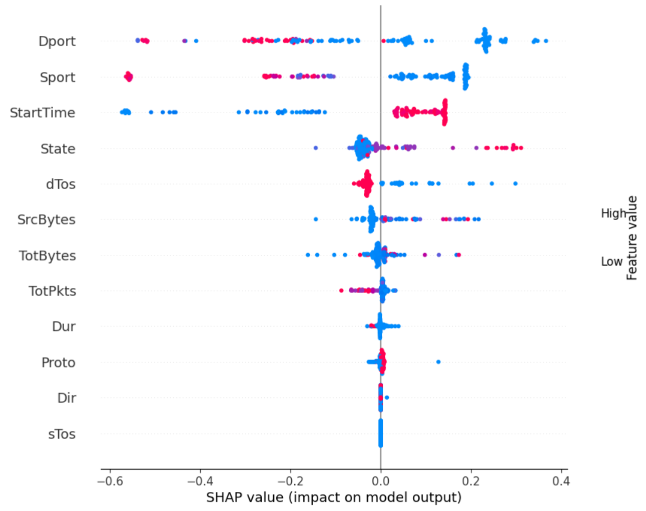
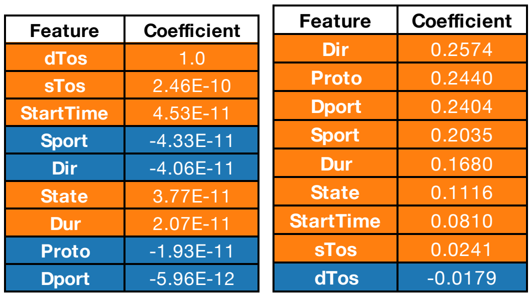


Figure A.2. Global SHAP summary plot for the GBM.



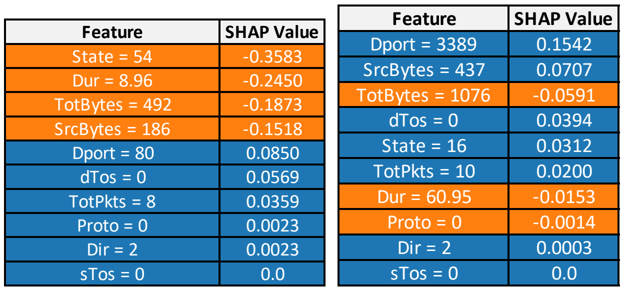
Feature	Coefficient
dTos	1.0
sTos	2.46E-10
StartTime	4.53E-11
Sport	-4.33E-11
Dir	-4.06E-11
State	3.77E-11
Dur	2.07E-11
Proto	-1.93E-11
Dport	-5.96E-12

Feature	Coefficient
Dir	0.2574
Proto	0.2440
Dport	0.2404
Sport	0.2035
Dur	0.1680
State	0.1116
StartTime	0.0810
sTos	0.0241
dTos	-0.0179

Figure A.3. LEMNA explanations with spurious features. (Left): False positive. (Right): False negative.

parser which feature fields should be read by means of providing a configuration file for the parser (2) The classifiers are implemented as a wrapper over the ML algorithms provided by `scikit-learn`. We currently support decision trees, logistic regression, explainable boosting machine, random forests, gradient boosting machine, and SVM. The wrapper specifies the ML algorithm and its hyperparameters. (3) Similarly, the explainers are also implemented as wrapper functions and currently provide support for SHAP, LIME, LEMNA, and ELI5. The modules can be extended for added support of custom parsers, models and explainers. For the sake of reproducibility, the pipeline saves the model, predictions and explanations in a file.

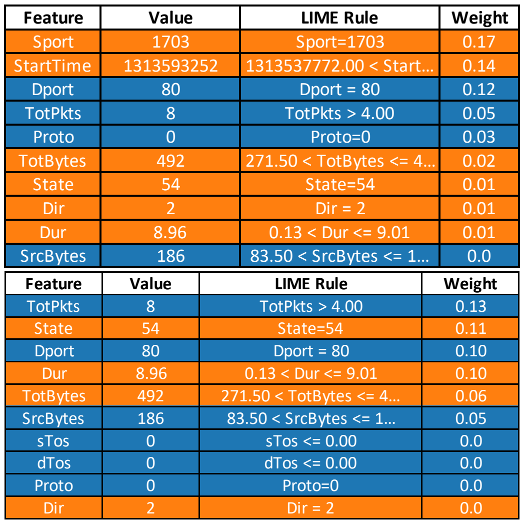
LEMNA implementation. We based our implementation of LEMNA on the code by Warnecke *et al.* [18]. For the explanation generation, we use the following settings: $N = 500, K = 6, S = 10$. The values of N and K are based on the original LEMNA paper. We do not need fused Lasso since our features do not have a temporal structure. Therefore we set S to a high value, effectively turning off the fusing effect. We expect LEMNA to perform better on tabular data when using feature discretization. However, optimizing LEMNA is out of the scope of this work as we are only using existing methods for model debugging.



Feature	SHAP Value
State = 54	-0.3583
Dur = 8.96	-0.2450
TotBytes = 492	-0.1873
SrcBytes = 186	-0.1518
Dport = 80	0.0850
dTos = 0	0.0569
TotPkts = 8	0.0359
Proto = 0	0.0023
Dir = 2	0.0023
sTos = 0	0.0

Feature	SHAP Value
Dport = 3389	0.1542
SrcBytes = 437	0.0707
TotBytes = 1076	-0.0591
dTos = 0	0.0394
State = 16	0.0312
TotPkts = 10	0.0200
Dur = 60.95	-0.0153
Proto = 0	-0.0014
Dir = 2	0.0003
sTos = 0	0.0

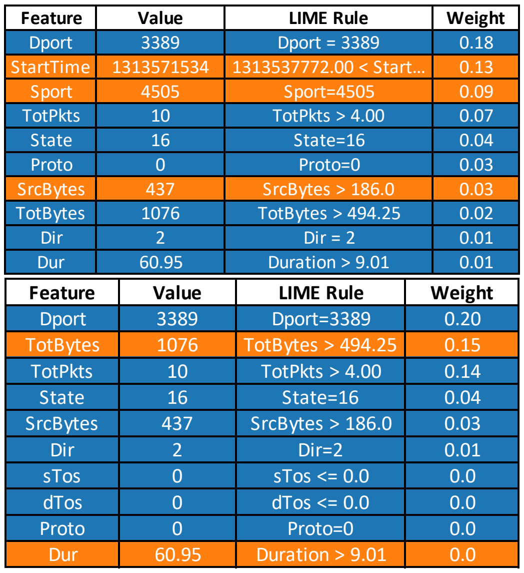
Figure A.4. SHAP explanations without spurious features. (Left): False positive. (Right): False negative.



Feature	Value	LIME Rule	Weight
Sport	1703	Sport=1703	0.17
StartTime	1313593252	1313537772.00 < Start...	0.14
Dport	80	Dport = 80	0.12
TotPkts	8	TotPkts > 4.00	0.05
Proto	0	Proto=0	0.03
TotBytes	492	271.50 < TotBytes <= 4...	0.02
State	54	State=54	0.01
Dir	2	Dir = 2	0.01
Dur	8.96	0.13 < Dur <= 9.01	0.01
SrcBytes	186	83.50 < SrcBytes <= 1...	0.0

Feature	Value	LIME Rule	Weight
TotPkts	8	TotPkts > 4.00	0.13
State	54	State=54	0.11
Dport	80	Dport = 80	0.10
Dur	8.96	0.13 < Dur <= 9.01	0.10
TotBytes	492	271.50 < TotBytes <= 4...	0.06
SrcBytes	186	83.50 < SrcBytes <= 1...	0.05
sTos	0	sTos <= 0.00	0.0
dTos	0	dTos <= 0.00	0.0
Proto	0	Proto=0	0.0
Dir	2	Dir = 2	0.0

Figure A.5. LIME explanations for false positive. (Top): With spurious features. (Bottom): Without spurious features.



Feature	Value	LIME Rule	Weight
Dport	3389	Dport = 3389	0.18
StartTime	1313571534	1313537772.00 < Start...	0.13
Sport	4505	Sport=4505	0.09
TotPkts	10	TotPkts > 4.00	0.07
State	16	State=16	0.04
Proto	0	Proto=0	0.03
SrcBytes	437	SrcBytes > 186.0	0.03
TotBytes	1076	TotBytes > 494.25	0.02
Dir	2	Dir = 2	0.01
Dur	60.95	Duration > 9.01	0.01

Feature	Value	LIME Rule	Weight
Dport	3389	Dport=3389	0.20
TotBytes	1076	TotBytes > 494.25	0.15
TotPkts	10	TotPkts > 4.00	0.14
State	16	State=16	0.04
SrcBytes	437	SrcBytes > 186.0	0.03
Dir	2	Dir=2	0.01
sTos	0	sTos <= 0.0	0.0
dTos	0	dTos <= 0.0	0.0
Proto	0	Proto=0	0.0
Dur	60.95	Duration > 9.01	0.0

Figure A.6. LIME explanations for false negative. (Top): With spurious features. (Bottom): Without spurious features.