This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2021.3117075, IEEE Transactions on Information Forensics and Security

Adversarial XAI methods in Cybersecurity

Aditya Kuppa, * Nhien-An Le-Khac *

School of Computer Science, University College Dublin, Ireland
Email: aditya.kuppa@ucdconnect.ie, an.lekhac@ucd.ie
*Corresponding author

***Abstract*—Machine Learning methods are playing a vital role in combating ever-evolving threats in the cybersecurity domain. Explanation methods that shed light on the decision process of black-box classifiers are one of the biggest drivers in the successful adoption of these models. Explaining predictions that address 'Why?/Why Not?' questions help users/stakeholders/analysts understand and accept the predicted outputs with confidence and build trust. Counterfactual explanations are gaining popularity as an alternative method to help users to not only understand the decisions of black-box models (why?) but also to provide a mechanism to highlight mutually exclusive data instances that would change the outcomes (why not?).**

Recent Explainable Artificial Intelligence literature has focused on three main areas : (a) creating and improving explainability methods that help users better understand how the internal of ML models work as well as their outputs; (b) attacks on interpreters with a white-box setting; (c) defining the relevant properties, metrics of explanations generated by models. Nevertheless, there is no thorough study of how the model explanations can introduce new attack surfaces to the underlying systems. A motivated adversary can leverage the information provided by explanations to launch membership inference, and model extraction attacks to compromise the overall privacy of the system. Similarly, explanations can also facilitate powerful evasion attacks such as poisoning and back door attacks.

In this paper, we cover this gap by tackling various cybersecurity properties and threat models related to counterfactual explanations. We propose a new black-box attack that leverages *Explainable Artificial Intelligence* (XAI) methods to compromise the confidentiality and privacy properties of underlying classifiers. We validate our approach with datasets and models used in the cyber security domain to demonstrate that our method achieves the attacker's goal under threat models which reflect the real-world settings.

***Index Terms*—XAI, Cybersecurity, Counterfactual Explanations, Adversarial Attacks, Poisoning Attacks, Model Stealing, Membership Inference Attacks.**

I. Introduction

Explainable Artificial Intelligence (XAI) is a multifacet discipline with influences from social sciences, philosophy, cognitive science, and psychology [1]–[3]. The field of explanations of intelligent systems was active in the 1970s mainly focused around expert systems; to, a decade after, neural networks; and then to recommendation systems in the 2000s [6].

The technical aspects of XAI methods can be grouped by when these methods are applied: before (pre-hoc), during (in-model), or after (post-hoc) building the machine learning (ML) model [4], [5]. Model explanations can be both *global* or *local*. A *global* explanation checks the inner workings of the whole ML model, by modeling the relationship between input and output spaces [10]. *Local* explanations try to interpret behind a decision/prediction of a single input data point (*test* sample), thus targeting a sub-region of the input space. Three main strategies for extracting explanations from ML models can be found in the literature: *domain-dependent*, *data-dependent* and *model-dependent*. Lately, methods leveraging optimization methods, causal and counterfactual inference [7]–[9] are gaining popularity in the literature.

With the successful deployment of XAI in real-world safety-critical systems [11], [12], [12]–[14], assessing the security, robustness, and reliability of underlying explanation methods is paramount for gaining adoption. Several metrics have been proposed in literature [15], [21] to measure the reliability, understandability, accuracy, and fidelity of the underlying XAI methods. Subjective metrics such as usefulness, completeness, and end-user satisfaction of a given explanation can be measured by surveying end users with a set of questions [42] or conducting controlled experiments. Similarly, biases in the explanation method are understood by measuring how distinctive/selective the method outputs are for different group/subgroup of inputs.

Very recently real-world XAI tests [18] are conducted on the fraud detection system in a Human-AI collaborative setting to evaluate the value of explanations generated by post-hoc methods. Authors observed that the decision accuracy worsens when an analyst is provided with Machine Learning (ML) model scores and explanations compared with data-only information. One can attribute this result to Fuzzy Trace Theory (FTT) [16] – an empirically validated theory of how humans interpret numerical stimuli. According to FTT, interpretability should be associated with less precise, yet productive and gist processing, whereas explainability may be more associated with the ability to understand the failure modes of the system via debugging than to use their output in real-world systems [17].

Real-world threat models to XAI systems can be categorized into:

- In a setting where explanations are legally required [23] manipulating the explanations may undermine the trustworthy evidence produced by these methods. In explanation manipulation attacks, a malicious model owner can leverage post-hoc explanation techniques to hide the weakness (fairness property) of the model and justify that the black-box model behaves fairly [19], [20]. Also, recent work has shown that explanations are sensitive to small perturbations of the input that do not change the classification result [14], [22].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2021.3117075, IEEE Transactions on Information Forensics and Security

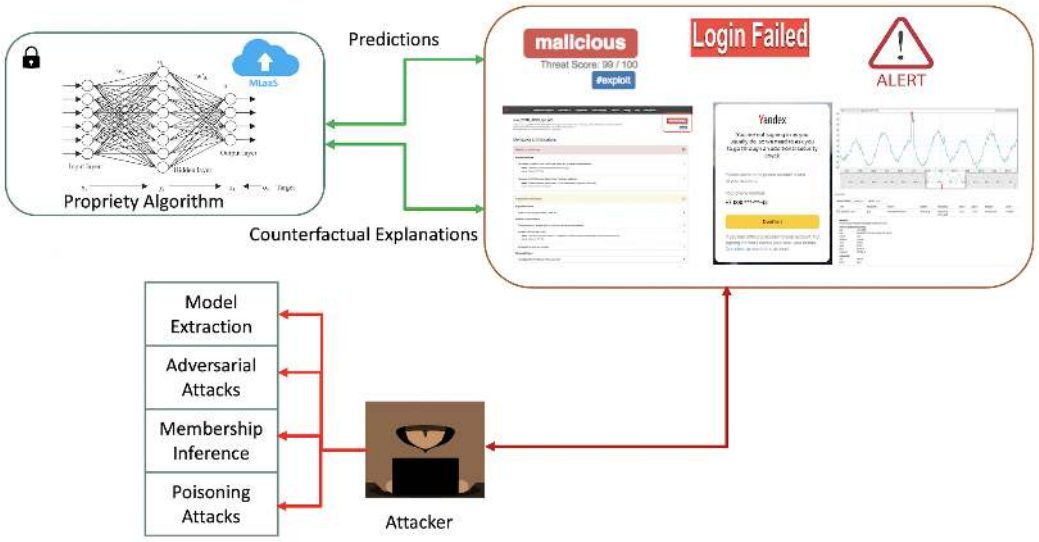


Fig. 1: Illustration of threats of real-world security systems which expose predictions and explanations to end-users; An anti-malware engine, which provides threat score with different properties of the file that were used to make the decision in the form of a report, an attacker can leverage explanations to tune the functionality of the file to bypass detection; A login authentication system which exposes masked phone numbers to third-party aggregators, a malicious attacker can run a membership inference attack to connect users with their phone numbers; A Network anomaly detection system reports the threshold and details about the attack in the report which can help an attacker to launch model stealing attacks on models shipped on edge devices.

- An Adversary compromising the security of the underlying system by leveraging explanations exposed to the system. These methods include Privacy compromises and Evasion attacks [44], [46]. Privacy degradation attacks are further categorized into model extraction and membership/attribute inference attacks. Evasion attacks include the generation of adversarial examples and, data/model poisoning, and backdoor injection techniques.

In the context of the cybersecurity domain, little work is done to understand the security robustness of explainable methods with a realistic threat model. Figure 1 illustrates different threats when a remote model provides explanation reports to end-users. Motivated by this, in this study we aim to conduct a security analysis of XAI methods, demonstrating how an adversary can use explanations to conduct evasion and privacy degradation attacks. More specifically, we seek to answer- (i) How can an attacker, given only outputs of explanation method and model predictions, can conduct powerful black-box model extraction, membership inference attacks? and, (ii) How explanation outputs facilitate the generation of adversarial samples and poison/backdoor samples to evade the underlying classifier? We first define the properties of the threat model for XAI methods into a unified attack framework and then conduct both analytical and qualitative studies of the security properties of these methods under realistic assumptions of the real-world adversary. The contribution of this paper can be listed as:

- We provide the first holistic security analysis of methods that exploit explanations, under real-world threat models.
- We propose a novel *black-box* attack, which leverages *XAI* methods to compromise confidentiality and privacy properties of underlying classifiers and show that our attack outperforms the relevant state of art methods in each attack category.
- Three cyber-security-relevant datasets and models are used to validate our approach to show that it achieves attacker goal under threat models which reflect the real-world settings.

II. Motivation

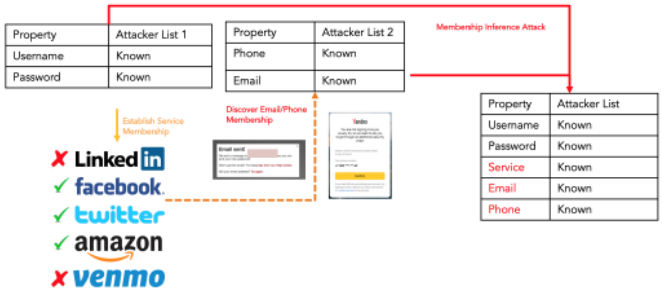


Fig. 2: Illustration of real-world Membership Inference attack (MIA); Attacker starts with sourcing two disjoint lists of leaked data, one with username, password and the other with email and phone numbers pairs. An attacker can run an MIA to establish a user account belongs to a service provider and can link email and phone numbers from the meta-data exposed by the service provider.

In this section, we cover our main motivation towards understanding the risk of counterfactual explanations in the context of cybersecurity use-cases.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2021.3117075, IEEE Transactions on Information Forensics and Security

Membership Inference attacks (MIA) The goal of a Membership Inference attack is to create a function that accurately predicts whether a data point belongs to the training set of the classifier or not [24], [25]. Recent work by [29] investigates the privacy risks of feature-based model explanations using MIA. They quantify information leakage of training data of the model based on its predictions and explanations.

Credentials stealing attacks can be formulated as MIA, in which an attacker aims to extract full information about a victim user from the data sourced from leaked databases to advance his/her attack campaign. Leaked data sources mostly contain only partial/incomplete information. Depending on the data available at hand, an attacker may rely on brute force, credential stuffing, and password spraying attacks to achieve his goal as shown in Figure 2. When an attacker sends a password reset request with a leaked email address, online services provide the attacker a "feedback" as to whether the email was valid or not with missing or contextual information. This meta-data may include masked phone numbers, usernames, etc. This information can be viewed as "what-if" and "why" explanations about login resets. Using this meta-data and partial data from leaked sources, an attacker can run an MIA to extract full information about the user.

Model Extraction attacks (MEA) Model Extraction [28] is the process where an adversary tries to steal a copy of an ML model, that may have been remotely deployed (such as over a prediction API). A recent survey [26] categorized different types of model stealing attacks into (i) Modification Attack in which an attacker fine-tunes the last layer of the stolen model by retraining with a new dataset and then compressing/pruning some layers to run the model on resource constraint devices. (ii) Active attacks are methods that tamper/modify the watermark/fingerprint of the model or modify the query to steal the functionality of the model. Leveraging gradient-based explanations for model extraction attacks was first demonstrated by Milli et.al [65].

In the cybersecurity domain, model stealing attacks are highly relevant due to the following reasons: (a) ML models are shipped to endpoints where security is limited. A large study [27] of AI mobile security applications indicates that most deep learning (DL) models are exposed without protection and can be easily extracted and pirated by attackers; (b) Adversary has a real motivation to steal the models to understand its internals, to bypass detection and test the attacks offline avoiding remote logging or alerting the owner; (c) In the field of cyber forensics, the remote verification of ML models is still in infancy. When a model exposes an explanation interface there is an increase in the attack surface, for example very recently authors [63], [89] used counterfactual explanations to steal the functionality of models under different threat model assumptions.

Poisoning attacks (PA) In a data poisoning attack, the attacker injects specially crafted data points into the training set such that the trained classifier predicted outputs can be influenced by attacker choice. Training data poisoning is mainly considered a threat when the system trains user-submitted/generated data in an incremental fashion. Depending on the attacker's goal, poisoning attacks can be executed to influence the prediction of a class [30] or a particular instance [31].

Adversarial Examples (AE) Inputs that are intentionally crafted to be in a close resemblance with benign samples but cause a misclassification by the underlying classifier. Depending on the threat model assumptions AE generation methods are categorised into three main classes: gradient estimation-based [32], [33], transferability-based [35], [36], [48], and local search-based methods [34]. Lately, researchers have explored explanation methods to defend models against AE-based attacks [37], but leveraging these methods to generate AE are slowly surfacing [14].

Explanations can play an important role in helping the adversary to improve his/her attack strategy for both poisoning and adversarial attacks. URL/File scanning/sandboxing[1] engines provide free services to check for a given file/URL is malicious or not for end-users. The output is often accompanied by a detailed report covering parameters, code blocks, and other metadata about the file/URL were used to score them. This content of the report can be viewed as explanations of the decision and can be misused by malicious attackers to tune their attack strategy of building malicious files. For example, if the entropy of a file is scored high, then the attacker can replace the obfuscation technique to reduce entropy and evade the detection.

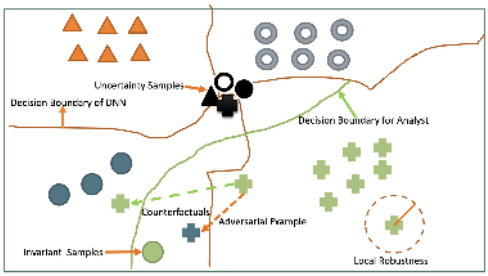


Fig. 3: Illustration of Decision Boundaries (DB) of human analyst and trained model. Depending on where the samples are placed we can divide them into 5 categories - (a) Locally robust which stay in some ϵ norm ball - Inside both human and model DB (b) Adversarial Samples [59] - Inside human DB but outside model DB (c) Counterfactual - Outside both human and model DB (d) Invariant examples [60] - Outside human DB but Inside model DB and, (e) Uncertainty samples - Both human and model DB are not well defined for these samples

Counterfactual Explanations Counterfactuals are human-friendly post-hoc local explanation methods, which address some of the bottlenecks of previous methods such as avoiding baselines, approximations to game theory constructs, and universality in features. Recently, Wachter et al. [52] proposed the use of counterfactual explanations in the context of GDPR [23] to help users to contest and understand model-based real/alternative decisions by asking "Why/Why Not" questions through counterfactuals. User-studies [41], [42] also showed

[1] https://www.joesandbox.com/

that users prefer counterfactual explanations when compared to feature importance methods.

Given an input instance to be explained, counterfactual data instances of the input have - (a) similar feature values as input (b) different model predictions from that of input (c) lay closer to the decision boundary of an input class. Depending on the counterfactual explanation generation algorithm, counterfactual methods can be categorized based on their -

- Access to model internals, gradients and prediction functions.
- Supports fully differentiable, linear, or piece-wise linear input models.
- Satisfy feasibility, sparsity, data manifold and, causality constraints.

We direct readers to recent surveys [53], [54] for a detailed analysis of different counterfactual methods found in the literature.

The astute reader may note that intuitively, counterfactual explanations (CF) generation methods share some similarities with Adversarial example (AE) generation methods in terms of how they leverage gradient-based optimization techniques and use of surrogate models for searching CF/AE for the target model. But they vary in a conceptual objective and the end goal, for example, AE aims to misclassify a sample to evade the target classifier whereas CF work towards finding data samples that not only have different predictions but also satisfy feasibility, sparsity, data manifold and, causality constraints. Similarly, AE's are used to understand the failure modes of underlying classifiers, whereas CF's help end-users understand the model decisions. Figure 3 summarizes different decision boundaries of human and learned models and differentiates between how CF/AE methods may be similar but satisfy different end goals. But one persona who is common for both AE and CF is a *malicious user/attacker* and he/she can launch attacks exploiting both the methods. This motivated us to explore the attacker's view of CF's i.e. how a motivated attacker can leverage CF's to achieve their goals.

III. Background

In this section, we will cover preliminaries and describe the threat model assumptions, attack definitions and, explanation methods. First, we describe the notations used in this work. We denote a scalar and a vector with a lowercase letter (e.g., t), and a boldface uppercase letter (e.g., $\mathbf{X}$), respectively. Table I shows a summary of notations that are frequently referred to and their problem context.

A. Attack Definitions

In a Membership Inference (MI) attack, the attacker's goal is to determine if a data sample (x) is a part of the training datasets of a target model $\mathcal{T}$. We formally define a MIA model $\mathcal{A}_{MemInf}$ as a binary classifier.

$$\mathcal{A}_{MemInf} : x, \mathcal{T} \mapsto \{member, non\text{-}member\} \tag{1}$$

In an attribute inference attack, the adversary's goal is to infer a specific sensitive attribute of a data sample from its representation generated by a target model. This sensitive attribute is not related to the target ML model's original classification task.

Given a data sample x and its representation from a target model, denoted by $h = f(x)$, attribute inference attack the adversary trains an attack model $\mathcal{A}_{AttInf}$ formally defined as follows:

$$\mathcal{A}_{AttInf} : h \mapsto s \tag{2}$$

where s represents the sensitive attribute.

In MEAs, the adversary goal is to steal the *functionality* of a victim model by training a surrogate model that is similar to the target model $\mathcal{T}$. Depending on the threat model assumptions, the success of the *functionality* replication is measured in terms of the surrogate model $\mathcal{A}_{MoExt}$ accuracy on target model test set $\mathcal{D}^{test}_{target}$.

$$\text{Acc}(\mathcal{A}_{MoExt}) = \frac{1}{|\mathcal{D}^{test}_{target}|} \sum_{x \in \mathcal{D}^{test}_{target}} \mathbb{I}(\mathcal{A}_{MoExt}(x) = \mathcal{T}(x)). \tag{3}$$

In DP, the adversary manipulates (add/update/delete) the training data in order to evade the classifier at the test time. More formally, the adversary adds m poisoning points $\mathcal{D}_{poison} = \{x_i, y_i\}_{i=1}^{m}$ into the target model training set $\mathcal{D}^{train}_{target}$, so that the learner minimises the poisoned objective $\ell(\mathcal{D}^{train}_{target} \cup \mathcal{D}_{poison}, \mathcal{T})$ rather than $\ell(\mathcal{D}_{target}, \mathcal{T})$. The poisoned set $\mathcal{D}_{poison}$ is constructed to achieve some adversarial objective $\mathcal{L}(\mathcal{T}(\mathcal{D}^{train}_{target} \cup \mathcal{D}_{poison}))$.

For generating adversarial samples, which evade the classifiers in the security domain, the attacker aims to manipulate a malicious sample without breaking its malicious functionalities, such that the underlying classifier misclassify it as benign. Unlike image domain counterparts, in security domain perturbations added to the sample have to preserve the functionality of the original sample. To satisfy the domain constraints, transformation functions $seq = \{a_1, a_2, \dots a_m\}$ from a predefined set A are used to modify the sample in the input space. More concretely, generating adversarial samples' problem can be viewed as optimisation problem with multiple objectives defined in Eq. 4

$$\begin{aligned} \min \quad & f_1 = P(\mathcal{T}(\boldsymbol{A_i} + \boldsymbol{X_m}) = \mathcal{T}(\boldsymbol{X_{mi}})) \\ \min \quad & f_2 = \|\boldsymbol{X_{mi}} - \boldsymbol{X_m}\|_{0,2,\infty} \\ \text{s.t.} \quad & A = \{a_1, a_2, \dots a_n\} \end{aligned} \tag{4}$$

where $P(\cdot)$ denotes the *confidence probability* of the classification result; $\boldsymbol{X_{mi}}$ and $\boldsymbol{X_m}$ represent the *adversarial perturbation* and *original malicious sample*, respectively; $seq = \{a_1, a_2, \dots a_m\}$ from transformation set A when applied to sample $\boldsymbol{X_m}$ preserves the functionalities of the original sample.

As shown in Eq. (4), the proposed multiobjective optimisation involves two objective functions. The first one f_1 represents the probability that the target model $\mathcal{T}$ classifies the generated adversarial example $\boldsymbol{A_i} + \boldsymbol{X_m}$ into the correct class. The remaining function are distance metrics, each of which is employed to evaluate the similarity between the original sample and the adversarial sample in feature space. The constraint imposed to $\boldsymbol{A_i}$ defines the range of perturbation functions based on the intrinsic property of the sample.

TABLE I: Notations

Symbol	Description	Problem Context
$\mathcal{T}$	Target model	Victim model which exposes inference and explanation interfaces
$\mathcal{D}_{target}^{train}$	Target Training Data	Victim model training data
$\mathcal{D}_{target}^{test}$	Target Test Data	Victim model test data
$\mathcal{D}_{target}$	Full dataset of Target model	$\mathcal{D}_{target}^{train} \cup \mathcal{D}_{target}^{test}$
$P(\cdot)$	Confidence Probability	Prediction probability of a given x used in the adversarial sample generation method
$\mathcal{T}(\cdot)$	Target Model Prediction API	Inference API of $\mathcal{T}$
$\mathcal{D}_{aux}$	auxiliary/surrogate dataset	Depending on the threat model assumptions an attacker can collect new data different from $\mathcal{D}_{target}$ distributions or from $\mathcal{D}_{target}^{train}$ distributions
$\mathcal{S}$	Local Attacker model	Shadow model build by an attacker using $\mathcal{D}_{aux}$
$\mathcal{E}_{explain}$	Explanation API	Depending on the counterfactual method this API outputs single/multiple data samples
$\mathcal{A}_{MemInf}$	Membership inference attack model	MI Surrogate model to predict the membership of a data sample x
$\mathcal{A}_{AttInf}$	Attribute inference attack Model	AI Surrogate model to understand the sensitive attribute s of a h of data sample x
$\mathcal{A}_{MoExt}$	Model extraction attack model	Surrogate model trained on counterfactual examples to extract the functionality of a victim model
$\mathcal{D}_{poison}$	Poisoning Dataset	Poisoning data injected into $\mathcal{D}_{target}^{train}$
A	Transformation Function	Domain specific functions which preserve the functionality of the sample when applied
x_{cf}	Counterfactual Example	Explanation produced by $\mathcal{E}_{explain}$ for sample x
$x_{cf1}, \ldots, x_{cfk}$	Diverse counterfactuals	Multiple diverse explanations produced by $\mathcal{E}_{explain}$ for sample x

B. Explanation Methods

For a given data point x to be explained, CF methods aim to find the data samples x_{cf} which have similar feature values but differ in target model predictions. This goal can be defined in terms of a distance function $Dist$ and a loss term $\mathcal{L}$ where $Dist$ minimizes the difference between an input and its CF in some input space and $\mathcal{L}$ function ensures that the predictions produced by a target model are different. More concretely,

$$x_{cf} = \underset{x_{cf1},\ldots,x_{cfk}}{\operatorname{argmin}} \; \mathcal{L}(\mathcal{T}(x_{cf}), \mathcal{T}(x)) + Dist(x_{cf} - x), \quad (5)$$

where k is the number of CF to be generated. In our work we use three different counterfactual methods in our experiments. Here we describe briefly each one. Table II summarizes each method loss function, optimisation algorithm and distance metrics.

Latent CF [38] uses the training set of a black-box classifier to train an autoencoder. To generate counterfactuals for a given data point, it perturbs the latent representation of sample $z = E(x)$ in the latent space until the desired class probability $f(D(z))$, is close to p. The final latent vector is fed into the decoder to generate the corresponding counterfactual.

Permute Attack [40] uses gradient-free optimisation technique based on the genetic algorithm to generate counterfactual. It leverages selection, crossover, and mutation steps to perturb the sample by permuting randomly selected features in an iterative fashion. The permutation values are selected randomly from possible values of the feature in the training data to make sure the chosen values are always valid and the probability distribution of features remains the same in the new generations. Hyperparameters mutation-range α, mutation probability ρ, population size d, and τ sampling temperature are tuned to generate counterfactuals.

$$\begin{aligned} \mathcal{L} &= \underset{x_{cf_1},\ldots,x_{cf_k}}{\operatorname{argmin}} \; \tfrac{1}{k}\Sigma_{i=1}^{k}\mathcal{L}(\mathcal{T}(x_{cf}),)\mathcal{T}(x) \\ proximity &= \tfrac{\lambda_1}{k}\Sigma_{i=1}^{k} Dist(x_{cf_i}, x) \\ diversity &= -\lambda_2 \,\mathrm{dpp\_diversity}(x_{cf_1}, \ldots, x_{cf_k}), \\ x_{cf} &= \mathcal{L} + proximity + diversity \end{aligned} \quad (6)$$

where λ_1 and λ_2 are the hyper parameters to tune the proximity and diversity of CF.

Diverse Counterfactual Explanations (DiCE) [39] generates multiple diverse CF's depending on the user input which are based on the diversity and proximity properties of CF's. Hinge-loss is used to make sure CF generated meets a minimum threshold of 0.5 between CF and the input class. Compared to standard ℓ_1 loss which finds examples closer to the input sample but maybe less feasible to the user, hinge-loss gives a flexible penalty term which ensures a zero penalty if the sample is above some threshold and a proportional penalty when the CF is below 0.5 threshold.

C. Threat Models

Several attempts have been made to categorize threat models for ML systems [43], [44], [46]. We distill the most important aspects that are relevant to our discussion in Table III.

Domain constraints In the cybersecurity domain, adversary has to respect the constraints i.e. given a feature vector of benign sample any modification (addition/removal/change) of features to achieve the attacker goal should preserve the original functionality of the sample in its parent domain. This is called the *inverse feature-mapping* problem and can result in multiple solutions. For example, simple malware classifier that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2021.3117075, IEEE Transactions on Information Forensics and Security

TABLE II: Counterfactual Explanation Methods leveraged in our work

Method	$\mathcal{L}$	$Dist$	**CF per** x	**Optimisation Method**
Latent CF [38]	Latent Vector Loss	ℓ_1	1	Gradient Descent
DICE [39]	Hinge-loss	ℓ_1 and Median Absolute Deviation(MAD)	k	Gradient Descent
Permute Attack [40]	-	l_2	1	Genetic Algorithm

TABLE III: Threat model assumptions for CF based attack

CHARACTERISTIC	TYPE	MEA [26]	MIA [68]	PA [67]	AE [45]
Knowledge	TRAINING DISTRIBUTION	✗	✗	✓	✗
	FEATURE SET	✓	✓	✓	✓
	FEATURE EXTRACTOR	✓	✓	✓	✓
	FEATURE TRANSFORMERS	✓	✓	✓	✓
	INFERENCE API	✓	✓	✓	✓
	EXPLANATIONS INTERFACE/METHOD	✓	✓	✓	✓
	CONFIDENCE INTERVALS	✓	✓	✓	✓
Goal/Intent	COMPROMISING INTEGRITY (EVASION)	✗	✗	✓	✓
	COMPROMISING PRIVACY	✓	✓	✗	✗
Capability	MANIPULATE TRAINING DATA	✗	✗	✓	✗
	MANIPULATE TEST DATA	✗	✓	✗	✓
Strategy	TRAIN A SURROGATE MODEL FOR PARAMETER EXTRACTION	✗	✓	✗	✗
	TRAIN A SURROGATE MODEL FOR QUERY REDUCTION	✓	✗	✗	✓
	SATISFY DOMAIN CONSTRAINTS	✗	✓	✓	✓
Frequency	ITERATIVE	✓	✓	✓	✓
Perturbation Scope	INSTANCE SPECIFIC	✓	✓	✓	✓
Perturbation Constraints	OPTIMISATION	✓	✗	✓	✗
	DOMAIN	✓	✓	✓	✓

uses byte code of the samples to train a n-gram classifier, an attacker can add small perturbations to the byte code fooling the classifier, but the perturbed file may not guarantee the functionality of the original malware sample. Also, the attack perturbation can be added to a feature or to the raw input. In our work for malware use-cases, we use transformation functions delete/modify/add $A = \{a_1, a_2, \dots a_m\}$ which when applied preserves the functionality of the file.

- Delete - Signer, Section, API calls and debugging information from the file
- Modify/Add - Signer, Section, API calls information , bytes to the end of file, NOP instructions such as, *mov eax, eax* and bogus code blocks, abstract syntax tree /control flow graph to insert dead nodes, API calls, Section names

We use Cuckoo sandbox[2] to verify the sample malicious functionality is preserved or not. The Cuckoo sandbox runs dynamic analysis on the sample and generates a report about all the actions performed by the sample after the execution. The report consists of malicious behaviours in a human-understandable explanation of the sample with a maliciousness score based on the behaviour. We consider a sample as malicious if its maliciousness score is higher than a threshold. For other datasets, we use feature-based perturbations only, so we did not impose any domain specific restrictions.

IV. PROPOSED ATTACK METHODOLOGY

In this section, we describe our attack methodology for compromising the privacy and confidence of the classifiers leveraging counterfactual explanations.

[2]https://cuckoosandbox.org/

Problem Setup. Given a black-box access to a target model $\mathcal{T}$, prediction interface $\mathcal{T}(x) = y$ and its counter factual explanation interface $\mathcal{E}_{explain}(x) = x_{cf}$, and an auxiliary dataset $\mathcal{D}_{aux}$ of $x_1, \dots, x_n$ the goal of the attacker is to compromise the confidentiality and integrity of the underlying ML system. Attacker can collect the $\mathcal{D}_{aux}$ from publicly available data and can reflect the $\mathcal{D}^{train}_{target}$ distributions or different distributions depending on the threat model assumptions. An attacker can send an inference request for a sample x to $\mathcal{T}$ prediction interface and it will return the prediction and the corresponding x_{cf} explanations. Depending on the explanation method employed $\mathcal{E}_{explain}(x)$ can serve k CF.

A. Privacy Attacks

Explanation-based model extraction To execute a successful MEA, an attacker has to take two things into consideration that can influence the success of the attack - (a) The $\mathcal{D}_{aux}$ should reflect $\mathcal{T}$ training set. It may happen that the collected data may not capture the training distributions, but given an input sample, the counterfactual explanation gives samples from different classes. An attacker can iteratively query different class samples to build the $\mathcal{D}_{aux}$ which capture the training set distributions. (b) Knowledge of target model architecture, a full knowledge can help to build a high accuracy/fidelity $\mathcal{S}$ which replicates the functionality of the victim model. In real-world threat models, the architecture of the victim model is not known to attackers, which makes the attack hard. To address this once we obtain data samples that reflect the training set, we employ the knowledge distillation technique to transfer knowledge from the target model to the surrogate model. More concretely, given the probability vector of target model $P_t(x)$

and shadow model $P_s(x)$ the distillation loss can be calculated by:

$$\mathcal{L}_{Distill}(\mathcal{T}, \mathcal{S}) = \mathcal{L}_{KL}(P_t(x), P_s(x)), \tag{7}$$

where $\mathcal{L}_{KL}$ indicates the KL divergence loss.

In our setup, the attacker first queries the victim model($\mathcal{T}$) with $\mathcal{D}_{aux}$ he has collected publicly and in turn trains a surrogate model $\mathcal{S}$ from the outputs provided by the Inference and explanation interface. Finally, we use the distillation loss in Eq. 7 to transfer knowledge from $\mathcal{T}$ to $\mathcal{S}$.

Explanation-based MI Attack Similar to the seminal work of Shokri et.al [24], we assume adversary can access the target model $\mathcal{T}$ in a black-box fashion. The $\mathcal{D}_{aux}$, does not come from the same distribution as the target model's training dataset. The auxiliary dataset and the counterfactual are used to train a shadow model $\mathcal{A}_{MemInf}$, the goal of which is to establish the membership evidence of a given sample.

Given a set of counterfactual examples x_{cfi} for input samples x_i of class y, we train a 1-nearest neighbour (*1-NN*) classifier that predicts the output class of any new input. The trained classifier predicts an instance closer to the CF examples as its counterfactual outcome class and instances closer to the original input will be classified as the original outcome class. We repeat the above process to train N 1-NN $\mathcal{A}_{MemInf}$ models one for each class in the dataset. For finding training data membership of a given data point, we compare the prediction probability between the $\mathcal{A}_{MemInf}$ and $\mathcal{T}$, if the difference is below threshold t we declare that the sample is part of the training set. The main intuition behind this method is if the target model and the counterfactual model both have the same prediction for a sample, that means that the sample should be influential for its own prediction. The advantage of this method is it does not need any access to the training set and uses CF examples of previous data points to build new data. The attacker can query the model in an iterative fashion to obtain new data.

B. Evasion Attacks

Explanation-based Poisoning Attack For a successful PA attack, one has to inject training samples, when trained on these samples can evade the classifier at test time for the attacker given inputs. In order to achieve this, the attacker has to first identify robust features which influence the classifier predictions, next he has to perturb the values of the identified features to sustain training loss. A trained poisoned model produces the correct output for a normal data sample x, $\mathcal{T}(x) = y$, and produces target class t as output for a poisoned sample x_p , $\mathcal{T}(x_p) = t$.

The first step in our approach is to identify robust features, which influence the class decision boundaries of the classifier. Once we have robust features we can perturb only these features for crafting poison samples instead of all the features. We observe the class-wise accuracy change by perturbing the features and filter out a subset of features based on their influence in prediction i.e. they are consistently same across their counterfactual class. The next step is to find the optimal value of the perturbation to make sure they achieve high training accuracy. To achieve this step, we solve the following optimization equation to minimize the distance between the poisoned sample x_p and a benign sample x in the input space.

$$\underset{x}{argmin} \, \|x_p - x\|_2^2$$

. Intuitively, this attack is similar to the poison frog attack [31], since counterfactual for a given sample already minimizes the sample distance in the feature space, we only have to work in the input space to find poison samples.

Explanation-based Adversarial Sample generation For Adversarial Samples, we leverage Permute attack as explanation API where counterfactuals are *adversarially generated*. Permute attack generates realistic counterfactual examples using permutation as the adversarial perturbation that keeps the range and the distribution of each individual feature the same as the original training data. Permute attack only works on the feature space, to support perturbation functions, which are constrained by A (Eq. 4), we modify the permutation $\Pi = \{\pi_1, \pi_2, \cdots, \pi_{n!}\}$ with $seq = \{a_1, a_2, \ldots a_m\}$ keeping the other dynamics same in the original algorithm.

V. Experimental Settings

In this section, we describe the datasets and experiment settings used to test the proposed attacks.

A. DataSets

Leaked Password Dataset. We use password-email pair data to test the counterfactual explanation-based membership inference attack. The dataset consists of 1.4 billion email password pairs with 1.1 unique emails and 463 million unique passwords. This dataset is aggregated password leaks from different incidents.

For a given leaked password and email address pair, the aim of the attacker is to discover other account pairs of the same user leveraging counterfactual explanations performing a MIA. To create the training set first we need to find a different set of usernames, emails, and passwords, which belong to the same user in the leaked list. We assume users sharing the same email address and username (the substring before @ of the email) but different/overlapping passwords belong to the same user, as it is a typical case of users having accounts in two different services connected via one email/usernames. Once we have email password pairs of the same user, we divide the email-password pairs by the email service provider i.e. we split the email 'alice@bob.com' into 'alice' and 'bob' and use 'bob' as the class label. The adversary goal is to discover other usernames/passwords for the same user with one leaked password and service provider as input.

Network Traffic. We use CICIDS17 [51] dataset for explanation based model extraction attacks. The network traffic dataset was collected in a controlled environment and contains network traffic in the packet-based and bidirectional flow-based format. For each flow, the authors extracted more than 80 features. The data set contains a wide range of attack types like SSH brute force, Botnet, DoS, DDoS, web, and infiltration

attacks. We use this dataset for the model stealing attacks. Features are extracted from bidirectional flows. Statistical time-related features are calculated separately for both directions. TCP flows are terminated by FIN packet and UDP flows are terminated by a flow timeout, which is set to 600 seconds. There are 8 groups of features that are extracted from raw pcaps: (a) Forward Inter Arrival Time, the time between two packets sent in forward direction (mean, min, max, std); (b) Backward Inter Arrival Time, the time between two packets sent backwards (mean, min, max, std); (c) Flow Inter Arrival Time, the time between two packets sent in either direction (mean, min, max, std); (d) Active-Idle Time, amount of time flow was idle before becoming active (mean, min, max, std) and amount of time flow was active before becoming active (mean, min, max, std); (e) Flags based features – Number of times the URG, PSH flags are set both forward and backward direction; (f) Flow characteristics – bytes per second , packets per second, length of flow (mean,min,max,std) and download and upload ratio of bytes; (g) Packet count with flags – FIN, SYN, RST, PUSH, ACK, URG, CWE and ECE; (h) Average number of bytes and packets sent in forward and backward direction in the initial window, bulk rate, and sub flows.

Malware. For testing explanation based poisoned and adversarial attacks, we collected a malicious sample dataset of 30120 malware from publicly available malware dataset virusShare[3] and for benign samples we scrapped 20334 clean files from free ware sites[4], [5], [6], [7]. We extract both raw and processed features for these binaries as described in EMBER [47].

B. Model Training

Model Extraction Target Model. An auto-encoder (AEnc) is trained on CICIDS data set with DDOS and benign attack classes. The auto-encoder network parameters such as (number of filters, filter size, strides) are chosen to be (53,10,1) for first and second layers and (53,10,1) for third and fourth layers of both encoder and decoder layers. The middle hidden layer size is set to be the same as rank $K = 20$ and the model is trained using Adam. Once the parameters are optimized after training, the AEnc model is used for anomaly detection, where an IP address and its time window are recognized as abnormal when the reconstruction error of its input features is high. Here, the reconstruction error is the mean square difference between the observed features and the expectation of their reconstruction as given by the AEnc. The threshold we used is 5% of the data as anomalies as this reflects the actual data set distribution. Once we have results from the anomaly detector, we train a random forest binary classifier on the output of the anomaly detector to make it a classification problem. We run the DICE explanation method in the black-box mode, for generating counterfactual explanations, meaning no feature scaling is done via median absolute deviation (MAD) features as all features are weighted equally in the normalized form. We sample 10k samples from the original dataset and use them as $\mathcal{D}_{shadow}$ and remove them from the dataset to train $\mathcal{T}$. We select time-based statistical features in the IDS dataset to ensure to respect the domain constraints in raw input space. The parameters to *desired_class* to "opposite", *proximity_weight* to 1.5 and *diversity_weight* to 1.0, *features_to_vary=['time_based_features]* and *feature_weights* by generating the MAD values from $\mathcal{D}_{shadow}$. We compare our model extraction attack with KnockoffNets (KN) [49] , and Knowledge distillation (KD) [50] attacks. KnockoffNets trains a stolen model via labeling surrogate dataset querying the victim model for predictions. We use $\mathcal{D}_{shadow}$ to train the stolen models, and query the target model with training samples of the shadow dataset. Next, we use the dataset built from these queries to train the stolen model a 3-layer Multi-Layer Perceptron (MLP) for 100 epochs using a Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.1. For KD we train each stolen model on the original training data, but with labels replaced by predicted probabilities from the target model.

Membership Inference Target Model. We first train an auto-encoder as a feature transformer to convert email password pairs into latent vector z of size 15. We use the latent vector to train a classifier with the service provider as labels and the latent vector as features. The network parameters (number of filters, filter size, strides) are chosen to be (50, 30, 15). Latent-CF is trained on a similar network as the target model for generating counterfactuals on $\mathcal{D}_{transfer}$. We sample 10000 email-password pairs to create $\mathcal{D}_{aux}$ dataset. We compare the attack with the *supervised learning-based approach* [24] and the *entropy-based approach* [55]. Both methods employ the shadow model training technique, the former trains a $\mathcal{S}$ on $\mathcal{D}_{aux}$ to check for the membership of a given data sample. The latter approach calculates a variant of the Shannon entropy of the prediction vector and determines the membership by checking whether this entropy value exceeds a certain threshold. For classification-based approach, we train 30 $\mathcal{S}$ 3-layer MLP for 100 epochs using an SGD optimiser with a learning rate of 0.1 on $\mathcal{D}_{aux}$ by varying training size to 1000, 2000, 3000, 4000, 5000, 6000, 8000, and 1,0000. For the membership classification, we train a binary classifier on logits and probability from $\mathcal{S}$. For entropy-based technique instead of a binary classifier, the threshold τ_y between member/non-member is learned with the shadow training technique [55].

Poison and Adversarial Attacks-Target Model. For poisoning attacks we train two target models (a) Gradient Boosting Model (GBM) similar to EMBER [47] (100 trees and 31 leaves per tree) as parameters; and (b) simple Neural Network (NN) based binary classifier 8 densely connected layers with Rectified Linear Unit (ReLU) activation with batch Normalisation and the final layer with Sigmoid activation. We sample 3% of test data from the malware dataset as $\mathcal{D}_{aux}$ for our attack. We filter out these samples from the $\mathcal{D}_{target}$ to reflect real-world threat model for the poisoning attack. The Permuteattack explanation method is applied on the $\mathcal{D}_{aux}$ with $\rho_0 = 0.4$

[3]https://virusshare.com/
[4]https://onlyfreewares.com/
[5]https://www.snapfiles.com/new/list-whatsnew.html
[6]https://downloadcrew.com/
[7]https://github.com/

and $\rho_1 = 0.1$, 100 generations, and mutation count to 20 with sampling temperature 0.3. For poisoning attacks, we aim to evade the GBM and NN models. Adversarial examples generated by our attack are tested on two commercial static anti-virus engines.

Evaluation Metrics. We adopt the attack accuracy as our evaluation metric for membership inference attacks following previous work [24]. Here accuracy means the success of the identification of users from two email service providers with respect to a number of counterfactual queries. For the model extraction attack, we use $\mathcal{S}$ accuracy on $\mathcal{T}$ test set $\mathcal{D}^{test}_{target}$. We measure **Attack Success Rate** (ASR) for the poisoning and adversarial example generation. This is measured by the accuracy of the model trained on poisoned data by the percentage of times a poisoned model is effectively tricked into misclassifying a *previously correctly recognized* malicious binary as benign. Similar to ASR of adversarial samples is measured by the evasion accuracy of the samples generated by permutate attack. We run our experiments on a server-grade machine with two IntelXeon E5-2640 v4 CPUs running at 2.40GHz, 64 GB memory, and Geforce GTX 1080 Ti GPU card.

VI. Results and Discussion

Counterfactual malware samples generated by our method, which evaded the commercial anti-virus engines employed simple changes to file. For example adding 1-4 bytes of debugging information or section name changes were the majority transformation functions employed by this attack Figure 4 (b) illustrates the transformations function counts in CF generated. Figure 4 (a) shows the evasion accuracy of the anti-virus engines. Our attack achieved evasion accuracy of 65% and 41% on two anti-virus engines under test. The functionality of the samples was preserved greater than 90%, which shows that counterfactual based adversarial sample generation method is useful in the wild. The results may highlight some of the weaknesses of the anti-virus engines but generally anti-virus engines use results from both static and dynamic analysis to make a decision. Our results are biased towards static features only, so we need to enhance our experiments to take into account dynamic features to test the robustness of the anti-virus engines. However, counterfactual explanation methods can help attackers to find quicker ways to find adversarial samples, instead of solving a hard-to converge black-box optimization problem in input space. Attackers can simply use counterfactual explanations to optimize their attack path. In Appendix Section **??** we investigate the influence of transformation functions on counterfactual sample generation methods.

Counterfactual based MIA on leaked passwords is a serious threat. This attack can help attackers to link accounts from various password leaks and improve their credential stuffing schemes. Password leaks due to mishaps are common in the real world and measures like cloaking and behavior-based restricting the login attempts are still some of the successful defensive methods. We measured the accuracy of linking usernames with passwords from a sample set of 1000 usernames. Other baseline methods such as shadow model-based and entropy-based attacks needed a large number of queries for a successful attack. We think the entropy-based method performed better than the state of art model-based method because of the difference in the distributions between training and testing. Searching the latent space to find the counterfactuals for a given sample worked because with a generative model in our case, AEnc learns similar samples to nearest neighborhoods [64]. In future we would like to explore how this attack can speed up the password cracking methods.

Counterfactual-based poisoned attacks were successful with the accuracy drop of the target model with a small percentage of poisoned samples in the training set, Figure 4 (d) illustrates the accuracy drop with poison sample percentage. We clearly see the correlation between increasing poison pool sizes to lower the accuracy of the target model. Transformation functions such as adding API calls and section information were employed in majority of the generated samples. We observed that the attack is successful at inducing targeted misclassification in the GBM and NN models. Poisoning the training pipeline of the security vendors is a major threat and in future we would like to explore how defenders can use counterfactuals to combat poisoning attacks.

Figure 4 (c) compares different attack models' accuracy with respect to the query count. Counterfactual-based model extraction attack was highly successful when compared to the other state of art methods. The main reason for this is methods like DICE optimizes to satisfy multiple properties like sparsity, proximity, diversity, and feasibility to generate counterfactuals. This optimization step helps the attacker to replicate the class-level decision boundaries of the target model by querying counterfactuals for each class. However, DICE does not support non-differential models and we aim to address this problem in future work. Table IV summarises all our experiment results and Table V gives the performance measures for each target model.

A. Efficiency and Complexity of CF attacks

We measure the computational efficiency of the proposed attack in terms of (a) latency – time taken by the method to generate the attack sample. (b) Sparsity – The changes made to the features by the method to generate the attack sample. We measure L_1 distance between the attack sample and original data for which CF was generated. Lesser the distance indicates a better choice of CF. Ideally inference calls to the original model for CF queries should be equal to the number of attack samples but not all CF generated are valid or satisfy domain constraints. Table VI summarizes the computational efficiency of the attacks proposed. Methods that query latent space for CF generation perform lesser inference calls. Gradient free methods based on genetic algorithms performed poorly in terms of latency due to underlying randomness in the search function.

The computational complexity of the attack depends on the number of the classes n, feature dimension d, number of queries q. For finding minimum perturbed attack samples using gradient-based methods, the loss function l computation for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2021.3117075, IEEE Transactions on Information Forensics and Security

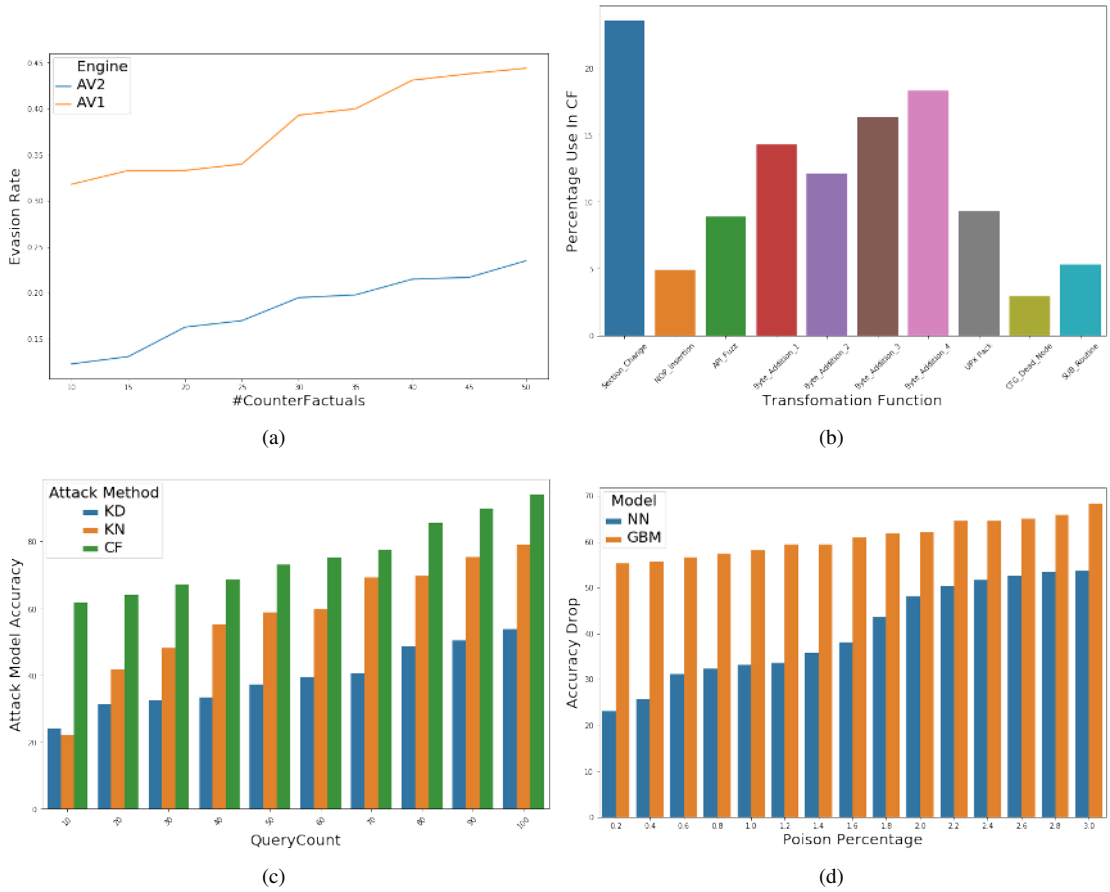


Fig. 4: (a) Adversarial attack on AV1 and AV2 anti virus engine: Evasion rate increases with counterfactual count but rate does not change with more than 50 counterfactuals which shows GA mutations exhausted the search space for transformation functions A. (b) Showcases the transformation functions applied in each generated CF, section change and byte additions are most used functions. (c) Model Extraction Attack: With increase in number of queries to $\mathcal{T}$ the attack accuracy increases for all the models but best accuracy comes from CF based attack. (d) Poisoning Attack: Accuracy drop of $\mathcal{T}$ is highly correlated with increase in percentage of poisoned samples in $\mathcal{D}_{target}$

each query influences the complexity of the attack. Similarly, for search-based methods the transformation functions F used in the mutations and the number of counterfactual k samples required to train the surrogate model play a significant role in the attack efficiency. In our CF-based attack, we apply KL-divergence loss for MEA and L_2 loss for poisoning attacks. For evasion and membership inference attacks we apply the genetic algorithm and nearest-neighbor-based search methods to achieve the attacker's goal. Table VII summarizes attack complexity of each attack type proposed.

B. Comparison with Baseline methods

We compare the proposed CF based attack with different state-of-the-art methods and report the performance.

1) Evasion: We compare the proposed CF-based evasion attack with two black-box anti-virus static evasion systems. In our experiments we utilize the soft label genetic programming-based black-box attack (GAMMA) [83] and Gym-malware [87] which uses OpenAI's gym environment to manipulate malware samples via reinforcement learning method. We measure the evasion rate by the ratio of the number of samples for which the label was flipped from malicious to benign. From Figure 5, we can see CF-based attacks performed better than the other methods for both black-box static anti-virus engines.

2) Poisoning: We compare the CF-based poisoning attack with two state-of-art poisoning methods namely, (a) Feature Collision (FC) attack [31], which crafts the poison samples by adding small perturbations to features so that the decision boundary lies closer to target sample; (b) Clean Label Poison attack [88], which adds the adversarial perturbation to each poisoned sample constrained by ℓ_∞-norm. We measure the test accuracy drop of the victim model with an increase in the percentage of training samples poisoned. From Figure 6, we can see CF-based poisoning attacks performed better than the other methods.

3) Membership Inference Attack: We compare the CF-based membership inference attack with the modified prediction entropy (MPE) attack [55] and the shadow models-based attack proposed by Shokri et.al. [24]. We measure the MIA model accuracy with an increase in the size of the training set to train surrogate models. Figure 7 shows the attack accuracy increase with more training data exposed to attackers. In the

ATTACK TYPE	$\mathcal{D}_{aux}$	EXPLANATION METHOD	$\mathcal{T}$	ORIGINAL ACCURACY	EVASION ACCURACY
ADVERSARIAL ATTACK	MALWARE	PERMUTE	AV1 AV2	93.5% 94.7%	65.23% 41.89%
				POISONING PERCENT	ACCURACY DROP
POISONING ATTACK	MALWARE	PERMUTE	GBM	0.5% 1% 2%	62.4% 76.23% 87.24%
			NN	0.5% 1% 2% 3%	30.9% 50.89% 65.31% 79.48%
				METHOD	ACCURACY/QUERIES
MEMBERSHIP INFERENCE	LEAKED PASSWORDS	LATENT-CF	AUTOENCODER	MODEL ENTROPY CF	49.46/1000 54.17/1000 73.17 /1000
				MODEL	ACCURACY
MODEL EXTRACTION	CICIDS	DICE	AUTOENCODER	$\mathcal{T}$ KN KD CF	98.02 78.91 53.89 93.54

TABLE IV: Summary of Experiment Results - Adversarial Attacks are measured by evasion rate of the counterfactuals generated by permute method. Functional column reports number of valid samples generated by CF and their evasion accuracy on two commercial anti-malware engines AV1, AV2. The success of the poisoning attack is measured by the accuracy drop of $\mathcal{T}$ when trained on poisoned CF's generated by permute method. MIA attack is compared with supervised and entropy-based methods. Success is measured by the Accuracy of identifying user-password pairs and the number of queries to the model for the identification of accounts. Model stealing attacks are compared with Knowledge distillation and KnockoffNets. The success of the attack is measured by the accuracy of the extracted model on $\mathcal{D}^{test}_{target}$. $\mathcal{D}_{aux}$ is generated by randomly sampling 3% of the whole $\mathcal{D}_{target}$ and discarded from it to reflect real-world setup.

Model	Dataset	Attack	Accuracy	Precision	Recall
GBM	Malware	Poisoning	0.975	0.991	0.998
AV1	Malware	Adversarial	99	-	-
AV2	Malware	Adversarial	98	-	-
AE	Password Leaks	Model Interference	0.967	0.931	0.965
AE	CICIDS	Model Extractions	0.9227	0.93770	0.9201
NN	Malware	Poisoning	0.980	0.926	0.924

TABLE V: Performance metrics of $\mathcal{T}$ on experiment datasets. For Adversarial attack we measure the detection accuracy of the av engine on the test set.

original experiments conducted by Song et.al [55], the MPE attack outperformed the shadow model attack but we did not see the high-performance gain in our experiments. It may be because if the confidence value distributions of train and test data are dissimilar then a single threshold value will not reflect the memberships.

4) Model Extraction Attack: We compare the proposed CF-based Model Extraction attack with Knowledge Distillation (KD) [50] and KnockoffNets [49]. KD methods help to transfer knowledge from one model (teacher) to another model (student) with similar accuracy as the teacher. It is widely popular in the model compression, network architecture search etc. Most knowledge distillation approaches require the knowledge of training data or teacher model intermediate weights, gradients, and activation statistics. Orekondy et al. proposed model stealing attacks that assumes access to a large dataset and use active learning to select the best samples to query [49]. We compare the accuracy of the stolen model on the test dataset of the victim model. Figure 8 summarizes the accuracy changes with the number of queries performed to train the stolen model. In Appendix Section **??** we evaluate the proposed CF attack on image and text data.

C. Defense Discussion

a) Defense Goals: An ideal *defense* method makes the underlying classifier robust, trustworthy, secure and aim to satisfy the following goals [69]

- **Low impact on the model architecture and accuracy:** when constructing any defense, one should aim for minimal changes to model architectures and classification metrics of the model.
- **Maintain model speed:** Defenses that are in line with the inference steps of the model should aim to maintain low latency to avoid degradation in response times.
- **Implementation Goals:**
  - Applicability across multiple model architectures, and domains.
  - Not tuned to a particular attack, threat model, or architecture.

TABLE VI: Computational Efficiency of counterfactual attacks. We measure the latency (in sec) of each method. L_1 distance between the original input and the CF gives the sparsity of the CF and number of calls to victim model gives the efficiency of the scheme, lesser the calls the more efficient the scheme is. We compute the mean of each metric for 100 randomly selected test points. We report the mean±std of this mean over 5 seeds.

METHOD	TIME IN SECONDS	AVERAGE DISTANCE (L_1)	CALLS TO ORACLE
	CF ALGORITHM/ATTACK TYPE		
DICE/MODEL EXTRACTION	75.48±1.98	0.029± 0.004	130± 1.9
PERMUTE/ADVERSARIAL EXAMPLES	132.23±5.82	0.124±0.009	280± 2.1
LATENT-CF/MEMBERSHIP INFERENCE ATTACKS	1.97±0.12	0.083±0.002	110± 2.4
PERMUTE/POISON	294.43±3.39	0.124±0.013	330± 1.5

ATTACK	COMPLEXITY
MEA	$MEA_a \sim \mathcal{O}(n * d * q^l)$
PA	$PA_a \sim \mathcal{O}(2 * d * q^l)$
EVASION	$EV_a \sim \mathcal{O}(n * F(log(d)))$
MIA	$MIA_a \sim \mathcal{O}(n * d * k)$

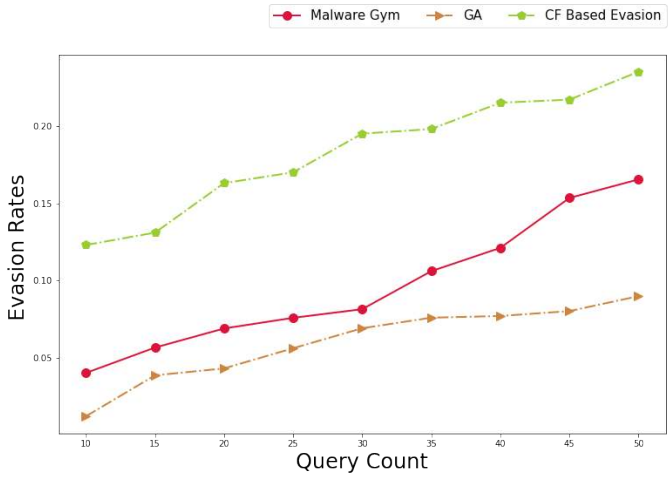
TABLE VII: Computational Complexity of CF-based Attack. n represents the number of classes of victim model, d the feature dimension, q^l average gradient updates of surrogate model per CF query, F the transformation functions applied in the genetic algorithm-based search, and k is the number of CF samples to train nearest neighbour surrogate models.

– Extensible, modular, and easily tunable to allow the designers to optimize on the utility-security trade-off based on the application.

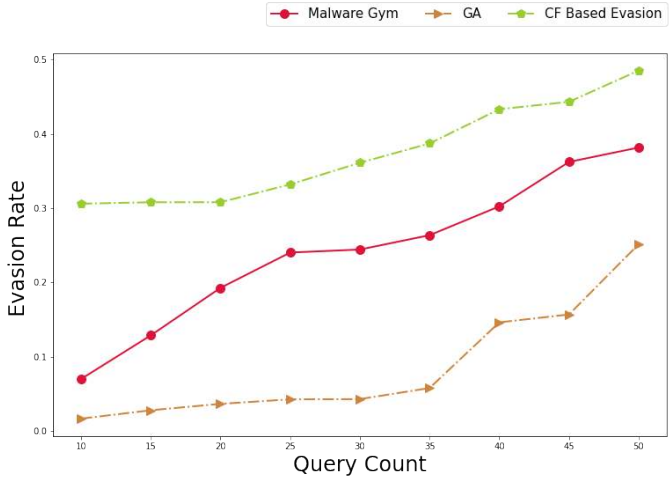
TABLE VIII: The advantages and disadvantages of defenses

TYPE	ADVANTAGES	DISADVANTAGES
MODIFIED TRAINING/INPUT	SIMPLE, GOOD DEFENSIVE ABILITY	DIFFICULT TO CONVERGE, HIGH OVERHEAD
MODIFYING THE NETWORK	LOW OVERHEAD, GOOD GENERALISATION	MODEL-DEPENDENT, HIGH COMPLEXITY
NETWORK ADD-ONS	LOW COMPLEXITY, MODEL-INDEPENDENT	WEAK GENERALIZATION, NOT IMPROVING THE ROBUSTNESS

As we can see from Table VIII there is no single category of defense that can support all the defense goals. The closest category is network add-ons i.e adversarial detection methods that have the least dependency on model training data, architecture, type of inputs, and assumptions about the attacker threat models. Also, most of the auxiliary defenses have detection modules attached to the network, which makes this method suitable for different domains.



(a) Anti-Virus Engine 1



(b) Anti-Virus Engine 1

Fig. 5: Evasion Results of Anti-Virus Engines

D. Potential Defenses for CF based Attacks

The process of generating counterfactual explanations shares a large set of similarities with adversarial examples concepts. For example, they both use similar distance metrics (L_0, L_2, and L_∞) to solve an optimization problem conditioned on some loss function. Given this resemblance, a potential direction towards defending CF-based attacks can be explored from adversarial defense literature. Adversarial training [70], [71] and input/network randomization [72]–[75] have proven to be most effective techniques against adversaries which employ first-order gradient-based optimisation techniques constraint by ℓ_p-norm bounds. CF methods that satisfy ℓ_p-norm constraint can adapt robust loss functions to restrict the CF samples lie in a small ϵ ball.

Other popular approaches in real-world scenarios – (a) Monitoring/Filtering of the incoming samples at inference

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2021.3117075, IEEE Transactions on Information Forensics and Security

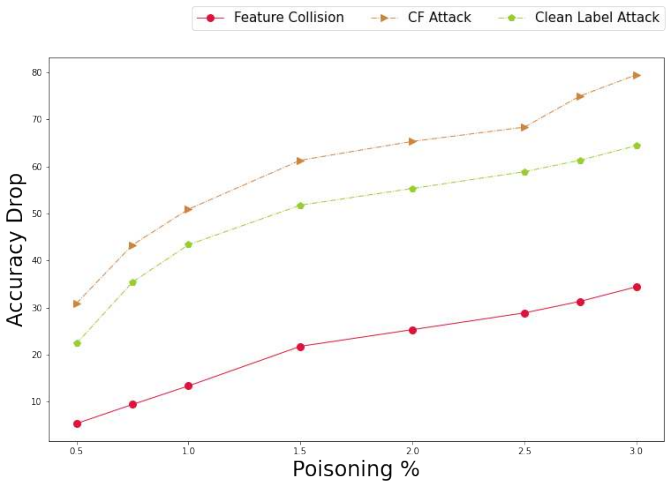


Fig. 6: Poisoning attack Results

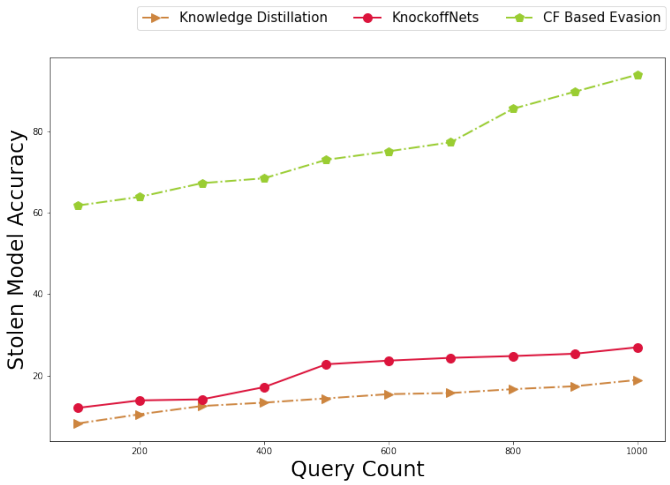


Fig. 8: Model Extraction attack Results

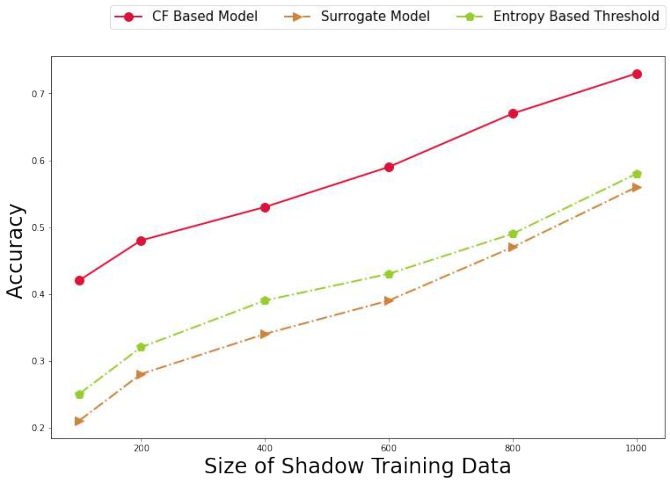


Fig. 7: MIA attack Results

time for behaviors which deviate from the clean sample via auxiliary detectors [78], [79]; (b) using statistical properties of clean samples to discover adversary injected samples [76], [77] and (c) active learning-based approaches with analysts in the loop can be explored to filter/identify attack samples.

Similarly, the choice of features used in the counterfactual generation process can play a vital role in defending the proposed attacks. Features that make the classifier output monotonically increasing have shown higher resistance towards injection attacks steered by gradient [80], content [82] and mimicry attacks [81]. An attacker modifying these monotonic features makes the sample more malicious to the classifier instead of benign. CF methods that employ feature-based distance methods to search for counterfactuals can leverage features that make the classifier prediction monotonic. Also, for a given sample x and its CF x_{cf} one can restrict the CF's feature values to lie in a very small neighborhood of class boundary. The distance can be measured in terms of largest change to any feature value, normalised by the standard deviation of that input feature $d(F(x), F(x_{cf})) = \max_i\{|x_i - x_{cf_i}|/\sigma_i\}$. This constraint can help defenders to flag attacker-guided CF search queries vs natural CF queries.

E. Noise based Defense

In our threat model, the attacker trains a surrogate model using the dataset collected from the public domain and queries the defender model for predictions and counterfactual samples to achieve his/her goal of evading the privacy/security of the system. In this scenario, the defender has no control over the attacker's *full* training data but only a portion of it (query response - counterfactual and predictions) used in training of the attack model. One strategy to combat CF-based attacks is – if the defender can transform the counterfactual samples in such a way that they reduce the accuracy of the trained surrogate model, then he can increase the attacker's budget making the attack hard if not impossible.

More formally, given an attacker collected dataset D_{aux} and corresponding counterfactual explanations CF_{aux} from model θ_D with D_{train}, D_{test} as its train and test dataset. Our aim is to design a transformation step T_s such that DNN trained on $T_s(CF_{\text{aux}})$ will perform poorly on D_{test}. The main intuition here is, generally, any ML model aims to learn the mapping function from the feature space to the label space from the training samples. So the T_s has to be designed in such a way that it induces noise into the CF_{aux} such that the learned model has a strong correlation between the labels and noise of the feature space instead of only features. This makes any learned model trained on $T_s(CF_{\text{aux}})$ effectively non-usable for attacker. Noise can be added for each counterfactual sample $x_{cf} + \delta_i$ or to all the samples of same class $x_{cf} + \delta_{y_i}$. Since most of the CF methods already search for samples constrained by ϵ norm, adding noise at class level can fool the attacker model. We design T_s similar to adversarial training but at the class level, adopting the first-order optimization method PGD [71]. For each example in class k, T_s applies $\boldsymbol{\delta}_k$ to the original example x_{cf} to produce x'_{cf}. The $\boldsymbol{\delta}_k$ accumulates over every example for the corresponding class k.

$$T_s = \begin{cases} None & \text{No transformation} \\ \phi, & \text{Random noise } [-1, 1] \\ \delta_{x_i}, & \text{Adv. noise } [-\epsilon, \epsilon] \\ \delta_{y_i} & \text{Adv. noise } [-\epsilon, \epsilon] \end{cases} \quad (8)$$

T_s	Accuracy
None	95.6
ϕ	87.4
δ_{x_i}	37.4
δ_{y_i}	28.22
$\delta_{x_i} + \phi$	32.22
$\delta_{y_i} + \phi$	18.22

TABLE IX: Accuracy drop of surrogate models with noise based defense

To test our hypothesis we train four models on CF_{aux} data set transformed by T_s function as per Equation 8. We report the drop in the accuracy of the model with respect to each transformation step. CF examples were generated using the DICE method for an MLP model trained on 10% of the CICIDS17 dataset. Table IX summarizes the accuracy drop of the models trained with different T_s transformations. We learn that adversarial noise added at per class level combined with random noise gives the highest drop in the attacker model accuracy. We plan to perform a detailed experimental study of the noise-based defense scheme in our future work.

F. Counterfactual Attack Games

Game theory can play a vital role in explaining and predicting various defense/offense strategies and designing effective threat models of security-sensitive systems [86]. In a game model, attackers and defenders are treated like players in a game and interactions between them are modeled as strategies, moves, and intent of opponents with respect to the utilities/actions of the players. Attacker and defender are expressed as min-max 2 player games where game dynamics, termination conditions are formulated as a Stackelberg/Nash equilibrium [84], [85] problem. Depending on the assumptions and constraints imposed on the system, as the game progresses a bi-level optimization problem, which is solved in terms of attacker/defender loss functions. One can view the CF-based attack setting as a min-max, max-min, or min-min game model. In Appendix Section **??** we study the proposed attack in the game theory settings. Finally, we aim to perform a detailed study in our future work.

VII. Related Work

For detailed coverage of each attack type, we refer readers to corresponding recent survey works [24], [26], [45], [61]. Here we cover methods that leverage explanations to achieve attackers' goal.

Model Stealing. Very recent works [63], [65] have started leveraging gradient-based and counterfactual explanations to execute model stealing attacks. Model stealing attacks in the cybersecurity domain are sparse and our work addresses this problem in real-world use-cases.

Model Inference Attacks. Leveraging explanation methods to execute an MIA is rare in literature and we address this gap in a real-world setting. Authors [29] leverage gradient-based explanations to perform membership and data set reconstruction attacks. In one of the experiments, they discuss an example-based explanation setup, for a given input sample the explanation method return samples similar to the input from the training set. Counterfactual examples can be viewed as a similar setup but with different outputs.

Poisoning Attacks. Recent works performed poisoning attacks through either polluting training data [31], [58] or modifying benign deep neural networks [56], [57]. Very recent work [62] similar to our work uses shapely values to perform poisoning and backdoor attacks on malware classifiers.

Adversarial Attacks. Demetrio [66] leverages integrated gradients to find the influential features for black-box decisions of an ML-based malware classifier. In [35], [36] authors demonstrated functionality preserving black-box attacks on network-based anomaly detectors and in [14] adversarial attacks were performed on explainable methods used in the security domain. In [89], a black box query-based attack was successfully performed on a face authentication system leveraging XAI techniques. In this work, we demonstrate functionality preserving adversarial examples that evade commercial antivirus systems leveraging counterfactual explanations.

VIII. Conclusion

In this work, we performed a detailed security analysis of models which expose counterfactual explanations. We design 4 black-box attacks that leverage *explainable artificial intelligence* (XAI) methods to compromise confidentiality and privacy properties of underlying classifiers. Leveraging 3 counterfactual explanation methods, we perform end-to-end evasion attacks on commercially available anti-virus engines, membership inference attacks to link users and discover their passwords from leaked datasets, and launch successful poisoning and model extraction attacks on real-world datasets and models. Through empirical and qualitative evaluation, we show the effectiveness of the attacks on varied datasets and highlight the security threats of exposing explanations to users and attackers alike.

Here we note directions for future work. First, we did not perform a detail study of CF based attack on different datatypes and model architectures. We plan to explore this direction in the future. Expand and improve the proposed attacks for different data types, model architectures and problem domains. Also, we aim to explore the application of counterfactual methods to defend against attacks similar to the one proposed in this work in future. As a final thought, there is always a tension between security and usability trade-offs in the cybersecurity domain that has been manifested in the field of explanations. On one hand, they can be excellent tools to explore and debug the model and data internals on the other hand, they may increase the attack surface of the system if access is not restricted.

REFERENCES

[1] Ruth MJ Byrne. 2019. Counterfactuals in Explainable Artificial Intelligence (XAI): Evidence from Human Reasoning.. In *IJCAI*. 6276–6282.

[2] Suresh Venkatasubramanian and Mark Alfano. 2020. The philosophical basis of algorithmic recourse. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. ACM.

[3] Solon Barocas, Andrew D Selbst, and Manish Raghavan. 2020. The hidden assumptions behind counterfactual explanations and principal reasons. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 80–89.

[4] Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608* (2017).

[5] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018b. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)* 51, 5 (2018), 1–42.

[6] Ashraf Abdul, Jo Vermeulen, Danding Wang, Brian Y Lim, and Mohan Kankanhalli. 2018. Trends and trajectories for explainable, accountable and intelligible systems: An hci research agenda. In *Proceedings of the 2018 CHI conference on human factors in computing systems*. 1–18.

[7] Raha Moraffah, Mansooreh Karami, Ruocheng Guo, Adrienne Raglin, and Huan Liu. 2020. Causal Interpretability for Machine Learning-Problems, Methods and Evaluation. *ACM SIGKDD Explorations Newsletter* 22, 1 (2020), 18–33.

[8] James Woodward. 2016. Causation and Manipulability. In *The Stanford Encyclopedia of Philosophy* (winter 2016 ed.), Edward N. Zalta (Ed.). Metaphysics Research Lab, Stanford University.

[9] Kuppa, Aditya and Aouad, Lamine and Le-Khac, Nhien-An Effect of Security Controls on Patching Window: A Causal Inference Based Approach, Annual Computer Security Applications Conference 2020

[10] Gill, P.H.N. Intr. to ML Interpre. ,O'Reilly Media, Incor 2018,

[11] N. Siddiqi, *Credit risk scorecards: developing and implementing intelligent credit scoring*. John Wiley & Sons, 2012, vol. 3.

[12] J. Kleinberg, H. Lakkaraju, J. Leskovec, J. Ludwig, and S. Mullainathan, "Human decisions and machine predictions," *The quarterly journal of economics*, vol. 133, no. 1, pp. 237–293, 2017.

[13] Miller, Claire Cain Can an algorithm hire better than a human, The New York Times, Volume 25, Jun 2015

[14] A. Kuppa and N. -A. Le-Khac, Black Box Attacks on Explainable Artificial Intelligence(XAI) methods in Cyber Security 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 2020, pp. 1-8, doi: 10.1109/IJCNN48605.2020.9206780.

[15] Carvalho, Diogo V and Pereira, Eduardo M and Cardoso, Jaime S Machine Learning Interpretability: A Survey on Methods and Metrics Electronics 8 2019, Multidisciplinary Digital Publishing Institute

[16] Valerie F Reyna. A new intuitionism: Meaning, memory, and development in fuzzy-trace theory. *Judgment and Decision making*, 2012.

[17] Gleaves, Lydia P and Schwartz, Reva and Broniatowski, David A The Role of Individual User Differences in Interpretable and Explainable Machine Learning Systems, arXiv preprint arXiv:2009.06675 ,2020

[18] Jesus, Sérgio and Belém, Catarina and Balayan, Vladimir and Bento, João and Saleiro, Pedro and Bizarro, Pedro and Gama, João How can I choose an explainer? An Application-grounded Evaluation of Post-hoc Explanations, arXiv preprint arXiv:2101.08758 , 2021

[19] Ulrich Aïvodji, Hiromi Arai, Olivier Fortineau, Sébastien Gambs, Satoshi Hara, and Alain Tapp. 2019. Fairwashing: the risk of rationalization. *arXiv preprint arXiv:1901.09749* (2019).

[20] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. 2019b. The dangers of post-hoc interpretability: Unjustified counterfactual explanations. *arXiv preprint arXiv:1907.09294* (2019).

[21] Arrieta, Alejandro Barredo and Díaz-Rodríguez, Natalia and Del Ser, Javier and Bennetot, Adrien and Tabik, Siham and Barbado, Alberto and García, Salvador and Gil-López, Sergio and Molina, Daniel and Benjamins, Richard and others Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI , Information Fusion, Elsevier 2020

[22] X. Zhang, N. Wang, H. Shen, S. Ji, X. Luo, and T. Wang, "Interpretable deep learning under fire," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020.

[23] European Commission (2016). *Regulation (EU) 2016/679: General Data Protection Regulation (GDPR)*.

[24] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 3–18. IEEE, 2017.

[25] Kuppa A., Aouad L., Le-Khac NA. Towards Improving Privacy of Synthetic DataSets. In: Gruschka N., Antunes L.F.C., Rannenberg K., Drogkaris P. (eds) Privacy Technologies and Policy. APF 2021. Lecture Notes in Computer Science, vol 12703. Springer, Cham.

[26] Xue, Mingfu and He, Can and Wang, Jian and Liu, Weiqiang DNN Intellectual Property Protection: Taxonomy, Methods, Attack Resistance, and Evaluations, arXiv preprint arXiv:2011.13564 2020

[27] A first look at deep learning apps on smartphones, Xu, Mengwei and Liu, Jiawei and Liu, Yuanqiang and Lin, Felix Xiaozhu and Liu, Yunxin and Liu, Xuanzhe The World Wide Web Conference , 2125–2136 2019

[28] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, pages 601–618. USENIX Association, 2016.

[29] Reza Shokri, Martin Strobel, and Yair Zick. 2019. Privacy risks of explaining machine learning models. *arXiv preprint arXiv:1907.00164* (2019).

[30] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 19–35. IEEE, 2018.

[31] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, pages 6103–6113, 2018.

[32] P. Chen, H. Zhang, Y. Sharma, J. Yi, and C. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *ACM Workshop on Artificial Intelligence and Security*, 2017, pp. 15–26.

[33] Y. Du, M. Fang, J. Yi, J. Cheng, and D. Tao, "Towards query efficient black-box attacks: An input-free perspective," in *ACM Workshop on Artificial Intelligence and Security*, 2018, pp. 13–24.

[34] M. Alzantot, Y. Sharma, S. Chakraborty, H. Zhang, C.-J. Hsieh, and M. B. Srivastava, "Genattack: Practical black-box attacks with gradient-free optimization," in *Genetic and Evolutionary Computation Conference*, 2019, pp. 1111–1119.

[35] Kuppa, Aditya and Grzonkowski, Slawomir and Le-Khac, Nhien-An Enabling Trust in Deep Learning Models: A Digital Forensics Case Study, 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), 2018, pages-1250-1255

[36] Aditya Kuppa, Slawomir Grzonkowski, Muhammad Rizwan Asghar, and Nhien-An Le-Khac. Black Box Attacks on Deep Anomaly Detectors In Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES '20119).

[37] Fidel, Gil and Bitton, Ron and Shabtai, Asaf When explainability meets adversarial learning: Detecting adversarial examples using SHAP signatures, 2020 International Joint Conference on Neural Networks (IJCNN)

[38] Balasubramanian, Rachana and Sharpe, Samuel and Barr, Brian and Wittenbach, Jason and Bruss, C Bayan Latent-CF: A Simple Baseline for Reverse Counterfactual Explanations , arXiv preprint arXiv:2012.09301 2020

[39] R. K. Mothilal, A. Sharma, and C. Tan, "Explaining machine learning classifiers through diverse counterfactual explanations," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 2020, pp. 607–617.

[40] Hashemi, Masoud and Fathi, Ali PermuteAttack: Counterfactual Explanation of Machine Learning Credit Scorecards, arXiv preprint arXiv:2008.10138 2020

[41] Reuben Binns, Max Van Kleek, Michael Veale, Ulrik Lyngs, Jun Zhao, and Nigel Shadbolt. 2018. 'It's Reducing a Human Being to a Percentage': Perceptions of Justice in Algorithmic Decisions. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3173574.3173951

[42] Jonathan Dodge, Q. Vera Liao, Yunfeng Zhang, Rachel K. E. Bellamy, and Casey Dugan. 2019. Explaining Models: An Empirical Study of How Explanations Impact Fairness Judgment. In *Proceedings of the 24th International Conference on Intelligent User Interfaces (IUI '19)*. Association for Computing Machinery, New York, NY, USA, 275–285. https://doi.org/10.1145/3301275.3302310

[43] Huang, Ling and Joseph, Anthony D and Nelson, Blaine and Rubinstein, Benjamin IP and Tygar, JD Adversarial machine learning Proceedings of the 4th ACM workshop on Security and artificial intelligence, 2011

[44] Papernot, Nicolas and McDaniel, Patrick and Sinha, Arunesh and Wellman, Michael Towards the science of security and privacy in machine learning ,arXiv preprint arXiv:1611.03814 ,2016

[45] Rosenberg, Ihai and Shabtai, Asaf and Elovici, Yuval and Rokach, Lior Adversarial Learning in the Cyber Security Domain, arXiv preprint arXiv:2007.02407, 2020

[46] Biggio, Battista and Roli, Fabio Wild patterns: Ten years after the rise of adversarial machine learning, Pattern Recognition ,2018

[47] Hyrum S Anderson and Phil Roth. Ember: an open dataset for training static pe malware machine learning models. *arXiv preprint arXiv:1804.04637*, 2018.

[48] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519. ACM, 2017.

[49] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[50] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[51] Sharafaldin, Iman and Lashkari, Arash Habibi and Ghorbani, Ali A Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization, ICISSP 108–116 2018

[52] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the gdpr," *Harv. JL & Tech.*, vol. 31, p. 841, 2017.

[53] Verma, Sahil and Dickerson, John and Hines, Keegan Counterfactual Explanations for Machine Learning: A Review arXiv preprint arXiv:2010.10596 2020

[54] C. Molnar, *Interpretable Machine Learning*, 2019, https://christophm.github.io/interpretable-ml-book/.

[55] L. Song and P. Mittal. Systematic evaluation of privacy risks of machine learning models. *CoRR*, abs/2003.10595, 2020.

[56] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning Attack on Neural Networks. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*.

[57] Yujie Ji, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. 2018. Model-Reuse Attacks on Deep Learning Systems. In *Proceedings of ACM SAC Conference on Computer and Communications (CCS)*.

[58] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *ArXiv e-prints* (2017).

[59] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[60] Jörn-Henrik Jacobsen, Jens Behrmann, Nicholas Carlini, Florian Tramèr, and Nicolas Papernot. Exploiting excessive invariance caused by norm-bounded adversarial robustness, 2020.

[61] Goldblum, Micah and Tsipras, Dimitris and Xie, Chulin and Chen, Xinyun and Schwarzschild, Avi and Song, Dawn and Madry, Aleksander and Li, Bo and Goldstein, Tom Data Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses, arXiv preprint arXiv:2012.10544, 2020

[62] Severi, Giorgio and Meyer, Jim and Coull, Scott and Oprea, Alina Exploring backdoor poisoning attacks against malware classifiers, arXiv preprint arXiv:2003.01031,2020

[63] Aïvodji, Ulrich and Bolot, Alexandre and Gambs, Sébastien Model extraction from counterfactual explanations , arXiv preprint arXiv:2009.01884 2020

[64] Hitaj, Briland and Gasti, Paolo and Ateniese, Giuseppe and Perez-Cruz, Fernando
Passgan: A deep learning approach for password guessing newblockInternational Conference on Applied Cryptography and Network Security , 217–237 ,2019 Springer

[65] S. Milli, L. Schmidt, A. D. Dragan, and M. Hardt, "Model reconstruction from model explanations," in *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 2019, pp. 1–9.

[66] Luca Demetrio, Battista Biggio, Giovanni Lagorio, Fabio Roli, and Alessandro Armando. Explaining vulnerabilities of deep learning to adversarial malware binaries. *arXiv preprint arXiv:1901.03583*, 2019.

[67] Goldblum, Micah and Tsipras, Dimitris and Xie, Chulin and Chen, Xinyun and Schwarzschild, Avi and Song, Dawn and Madry, Aleksander and Li, Bo and Goldstein, Tom Dataset Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses, arXiv preprint arXiv:2012.10544 2020

[68] Hu, Hongsheng and Salcic, Zoran and Dobbie, Gillian and Zhang, Xuyun Membership Inference Attacks on Machine Learning: A Survey, arXiv preprint arXiv:2103.07853 2021

[69] Aldahdooh, Ahmed and Hamidouche, Wassim and Fezza, Sid Ahmed and Deforges, Olivier Adversarial Example Detection for DNN Models: A Review, arXiv preprint arXiv:2105.00203,2021

[70] Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[71] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[72] Raff, E., Sylvester, J., Forsyth, S., and McLean, M. Barrage of random transforms for adversarially robust defense. In *CVPR*, 2019.

[73] Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. In *ICML*, 2019.

[74] Pang*, T., Xu*, K., and Zhu, J. Mixup inference: Better exploiting mixup to defend adversarial attacks. In *ICLR*, 2020.

[75] Dhillon, G. S., Azizzadenesheli, K., Bernstein, J. D., Kossaifi, J., Khanna, A., Lipton, Z. C., and Anandkumar, A. Stochastic activation pruning for robust adversarial defense. In *ICLR*, 2018.

[76] Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019.

[77] Zhihao Zheng and Pengyu Hong. Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 7913–7922, 2018.

[78] Tianyu Pang, Chao Du, Yinpeng Dong, and Jun Zhu. Towards robust detection of adversarial examples. In *Advances in Neural Information Processing Systems*, pages 4579–4589, 2018.

[79] Fatemeh Sheikholeslami, Swayambhoo Jain, and Georgios B. Giannakis. Minimum uncertainty based detection of adversaries in deep neural networks. In *Information Theory and Applications Workshop, ITA 2020, San Diego, CA, USA, February 2-7, 2020*, pages 1–16. IEEE, 2020.

[80] Inigo Incer, Michael Theodorides, Sadia Afroz, and David Wagner. 2018. Adversarially Robust Malware Detection Using Monotonic Classification. In *Proceedings of the Fourth ACM International Workshop on Security and Privacy Analytics*. ACM, 54–63.

[81] David Wagner and Paolo Soto. 2002. Mimicry attacks on host-based intrusion detection systems. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*. ACM, 255–264.

[82] Antoine Wehenkel and Gilles Louppe. 2019. Unconstrained monotonic neural networks. In *Advances in Neural Information Processing Systems*.

[83] Demetrio, Luca and Biggio, Battista Secml-malware: A Python library for adversarial robustness evaluation of windows malware classifiers, arXiv preprint arXiv:2104.12848 ,2021

[84] Michael Brückner, Christian Kanzow, and Tobias Scheffer. Static prediction games for adversarial learning problems. *Journal of Machine Learning Research*, 13:2617–2654, 2012.

[85] Brückner, M. and Scheffer, T. Stackelberg games for adversarial prediction problems. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pp. 547–555, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450308137. 10.1145/2020408.2020495.

[86] Wang, Yuan and Wang, Yongjun and Liu, Jing and Huang, Zhijian and Xie, Peidai A survey of game theoretic methods for cyber security, 2016 IEEE First International Conference on Data Science in Cyberspace (DSC)

[87] Anderson, Hyrum S and Kharkar, Anant and Filar, Bobby and Roth, Phil Evading machine learning malware detection, Black Hat, 2017

[88] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Clean-label backdoor attacks. 2018.

[89] Garcia, Washington and Choi, Joseph I and Adari, Suman K and Jha, Somesh and Butler, Kevin RB Explainable black-box attacks against model-based authentication, arXiv preprint arXiv:1810.00024, 2018