"Real Attackers Don't Compute Gradients": Bridging the Gap Between Adversarial ML Research and Practice

Giovanni Apruzzese*, Hyrum S. Anderson§, Savino Dambra¶, David Freeman†, Fabio Pierazzi‖, Kevin Roundy¶
*University of Liechtenstein, §Robust Intelligence, ¶Norton Research Group, †Meta, ‖King's College London
{name.surname}@{uni.li*, nortonlifelock.com¶, kcl.ac.uk‖}, dfreeman@meta.com†, hyrum@robustintelligence.com§

***Abstract*—Recent years have seen a proliferation of research on *adversarial machine learning*. Numerous papers demonstrate powerful algorithmic attacks against a wide variety of machine learning (ML) models, and numerous other papers propose defenses that can withstand most attacks. However, abundant real-world evidence suggests that actual attackers use simple tactics to subvert ML-driven systems, and as a result security practitioners have not prioritized adversarial ML defenses.**

Motivated by the apparent gap between researchers and practitioners, this position paper aims to bridge the two domains. We first present three real-world case studies from which we can glean practical insights unknown or neglected in research. Next we analyze all adversarial ML papers recently published in top security conferences, highlighting positive trends and blind spots. Finally, we state positions on precise and cost-driven threat modeling, collaboration between industry and academia, and reproducible research. We believe that our positions, if adopted, will increase the real-world impact of future endeavours in adversarial ML, bringing both researchers and practitioners closer to their shared goal of improving the security of ML systems.

***Index Terms*—Threat Model, Economics, Cybersecurity, Machine Learning, Research, Practice, Adversarial**

I. Introduction

Several recent surveys indicate that protecting machine learning (ML) models is not a leading security concern for practitioners [1]–[5]. According to the few recorded accounts of security failures "in the wild," ML systems can be broken by naïve attackers that are not systematically exploiting the vulnerabilities of ML, but rather are developing attacks by guessing—either indiscriminately or by some coarse heuristic [6], [7]. Red-team exercises on ML systems often take advantage of security gaps that are agnostic to the existence of an ML model, and subsequent defensive recommendations are likewise more broad than, e.g., *adversarial training* [8], [9]. Additionally, the ML models deployed in production-grade ML systems are often not directly observable (and are sometimes even unreachable) by most attackers [10].

On the other hand, researchers assert that real ML implementations should not follow the principle of "security by obscurity" [11], and that security evaluations of ML models should assume worst-case scenarios [12]–[14]. Indeed, no one can exclude the possibility of future powerful adversaries turning their attention to the ML models embedded in real systems. Fueled by such "what ifs," thousands of papers [15] have showcased successful security violations of ML models by means of sophisticated strategies such as gradient-based algorithms [16].

These few anecdotes suggest that **the field of ML security suffers from a mismatch between the priorities of practitioners and the focus of researchers.** In this position paper we argue that the gap between adversarial ML research and practice is real and identify several underlying causes. We aim to reduce this gap by proposing guidelines for future endeavours that reflect our observations of aspects overlooked by, or inconsistent within, existing literature.

To reach our goal, we first revisit the fundamentals of ML security (§II). We argue that real deployed ML models are part of complex ML *systems* whose architectures are unknown to researchers, and that cybersecurity is rooted in *economic* considerations for both attackers and defenders.

To assist researchers in understanding crucial facets of operational ML security, we then present three original case studies from the real-world (§III) that elucidate: (a) the *architecture* of a complex ML system deployed in an online social network; (b) the *lack of evidence* of adversarial examples against a commercial ML system for phishing webpage detection; and (c) the role of *time and domain expertise* in devising sophisticated attacks during an acclaimed ML evasion competition.

Next we turn our attention to the research domain and take a snapshot of the current landscape of adversarial ML as portrayed in scientific papers (§IV). After surveying the proceedings of the "Top-4" security conferences from 2019 to 2021, we systematically analyze all 88 papers that consider attacks against ML or corresponding defenses. Of these papers, 89% only evaluate algorithms based on neural networks, 63% focus on computer vision, and 80% perform their experiments on "benchmarks". We discover several inconsistencies in the terminology adopted in reputable prior work. We also identify several positive trends, such as an increasing amount of papers envisioning attackers who "ignore" the ML model and reach their goal by targeting other elements of the ML system.

Finally, we coalesce all our observations by stating four positions that researchers and practitioners in ML security can adopt to close the gap between these domains (§V). We encourage the community to (a) adapt threat models to ML systems, (b) integrate threat models with cost-driven assessments, (c) build collaborations between industry and academia, and (d) embrace a "just culture" for source-code disclosure.

In summary, we make three major contributions:

- We present three *real-world case studies*, showing practical insights unknown or neglected in research.
- We analyze all recent adversarial ML papers in top security venues, highlighting positive trends and blind spots.
- We state four positions that, if adopted, will help bridge the gap between research and practice in ML security.

Our contributions also include extended discussions (Appendix A) and the detailed findings of our literature review (Appendix B). Our resources can be found in our website [17].

II. Revisiting Machine Learning Security

We begin by reviewing the foundations of the three pillars of our paper: machine learning systems (§II-A), security of machine learning (§II-B), and practical cybersecurity (§II-C). We also compare our paper with related work (§II-D).

A. Overview of Machine Learning (Systems)

Machine learning in one paragraph. The fundamental goal of machine learning (ML) is to create "machines that can make decisions by learning from data" [18]. At its core, the process entails applying a *learning algorithm* to *training data*, which yields a *machine learning model*. The purpose of any ML model is to generalize to new data: given an *input*, the ML model produces an *output* that can be used to accomplish a given task, such as predicting future trends, detecting malware, recognizing speech patterns, or inferring objects in images [18], [19]. The effectiveness of an ML model is measured during a *validation phase*, during which the ML model analyzes test data and its predictions aer compared to some known *ground truth* (e.g., labels [20]). Effectiveness can be measured through various metrics (e.g., *accuracy*). Informally, an ML model belongs to either of two "paradigms" that characterize the underlying learning algorithm. Those based on neural networks [19] are denoted as *deep learning* (DL), whereas others are broadly denoted as *shallow learning* and include, e.g., SVMs and tree-based algorithms [21].

Using ML in practice. If an ML model is determined to be effective, it can be integrated into a *machine learning system*, which itself can be a component of some product or service. Indeed, an ML model by itself is usually insufficient for operational deployment. First, it must receive an *input* that may itself be derived from a separate component (e.g., signal capture or preprocessing[1]). Second, the ML model's *output* may be further analyzed for decision making. We illustrate an ML system with a linear data-processing pipeline in Fig. 1.

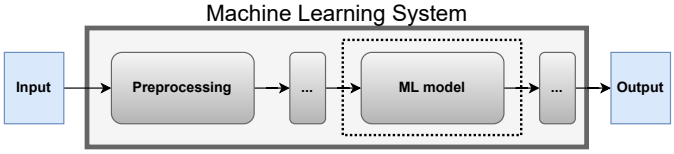


Fig. 1: An ML system. The system receives an input, which is preprocessed and then fed to an ML model, the results of which may be further processed before providing the system's final output.

[1]Preprocessing steps can be complex. They can entail, e.g., transformations, filtering, normalization, feature extraction, and/or noise reduction.

Real ML systems may include pipelines having various components, each potentially having its own applications of ML (e.g., image scaling [22], word embeddings [23]). Some ML systems require constant updates, which can be done in an automated fashion [24], while others may adopt company-specific guidelines [25]. It is even possible that a given input never reaches *any* ML model within the system. This outcome is typical, e.g., in Network Intrusion Detection Systems (NIDS), wherein samples matching known "signatures" are put into an alternate pipeline that does not include any ML [26].

> Observation: A machine learning **model** is merely a single component within a machine learning **system**.

Types of ML systems. Many IT infrastructures already rely on ML systems at either small or large scales. Without loss of generality we identify two main categories of ML systems, characterized by the relationship (i.e., knowledge and degree of interaction) between the ML system and its users.

- Open. The source code of these ML systems is publicly available. This category may include custom-built ML systems where the code is subsequently released. Obviously, any ML system is open for its developers.
- Closed. These ML systems are developed in a proprietary setting and distributed to their end users, who can inspect neither (i) the underlying source code, nor (ii) the components enclosed in the ML system. These ML systems are either *unrestricted*, if they can be freely used; or *restricted*, if their use is subject to some restriction, such as a fee or a threshold based on queries.

We provide some examples of these ML systems in Table I.

TABLE I: Types of ML systems with example products and research papers.

Type of ML system	Example Application		
	Product / Company	*Use case*	*Related Work*
Open	OpenPilot [27] GPT2 [29]	Autonomous Driving Natural Language	Sato [28] Carlini [30]
Closed	Google Translate [31] ClarifAI [33]	Translation (*unrestricted*) MLaaS (*restricted*)	Pajola [32] Yu [34]

Both open and closed ML systems resemble the schematic shown in Fig. 1: they receive an *input* and provide an *output*. Typically, the users of an ML system can control the input and can observe the output (e.g., via an API [35]). Such properties, however, may not hold for some special cases, which we denote as invisible ML systems. Control and configuration of invisible ML systems are reserved for "high-privilege" users (e.g., developers or sys-admins), for whom the ML system can be either open or closed. The functionality of these systems depends on the interactions of "low-privilege" users (e.g., employees or customers), to whom the operation of the ML system is not readily apparent. A low-privilege user may receive some feedback based on the ML system's output, but such feedback may be influenced by other systems or may arrive after a long and/or unpredictable delay. Simply put, low-privilege users of invisible ML systems may not know (i) **if** something happened, (ii) **when** it happened, (iii) **why** it happened, and/or (iv) **what** changed. We provide a schematic

of an INVISIBLE ML system for network management in Fig. 2 (two of our case studies in §III entail INVISIBLE ML systems).

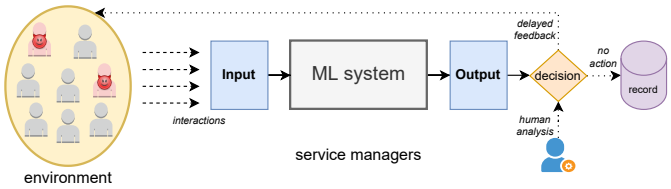


Fig. 2: An INVISIBLE ML system. The low privileged users in the *environment* interact with the ML system, the output of which can be used in diverse ways by its *managers*. For example, the alarms of a NIDS may be inspected by sysadmins and then used to block suspicious hosts; whereas a network management system can dynamically allocate available resources (e.g., bandwidth). In some cases the output does not trigger any immediate action.

B. Security of Machine Learning

Adversarial Machine Learning. Thousands of papers [15] have addressed the security of machine learning, a research field typically denoted as "adversarial machine learning" [36]. Although work in this field has existed for nearly 20 years [37], adversarial ML became popular when Szegedy et al [38] showed that deep learning is vulnerable to *adversarial examples*, which are examples that are "similar" to "normal" inputs but induce a given ML model to behave incorrectly [39].

Adversarial examples can certainly be used in malicious ways: for instance, an attacker can craft an adversarial example that *evades* an ML model powering a security system, e.g., a malware classifier [40]. (We provide the definition of evasion attacks in Appendix A-A.) Nevertheless, we will show that adversarial examples are only one of a myriad of threats that can violate the security of ML systems.

Threat Model. Security assessments require the definition of a *threat model*, which establishes the relationship between an attacker and their target [10]. The seminal paper by Huang et al. [36] was among the first to formalize the potential threats to ML. Since then, more efforts followed—some more general (e.g., [39], [41]), others more specific (e.g., [42], [43]). In the ML context, threat models characterize an attacker's relationship to an ML model through four elements: GOAL, KNOWLEDGE, CAPABILITIES, and STRATEGY.

GOAL. Attackers can have diverse GOALS, such as violating the *integrity* or the *availability* of the ML model [44]. They can also, however, attempt to "steal" the ML model [45], or extract private data from it via *membership inference* [46] attacks that infer whether a person's data is in the model's training set.

KNOWLEDGE. Srndiç and Laskov [43] pointed out that the attacker's KNOWLEDGE spans over three elements of an ML model: the training data, the feature set, and the algorithm. Depending on the information available to the attacker, most literature adopts a "box" terminology [41]: "white-box," when the attacker knows *everything*; "black-box," when the attacker knows *nothing*; and "gray-box," for intermediate cases [42].[2]

CAPABILITIES. Existing literature shows that attackers use various CAPABILITIES to tamper with ML models. For instance, they can interfere with the training phase of the ML model [47]; they can control inputs to a (trained) ML model and observe its output [48]; and/or they can operate from diverse spaces of the data pipeline of a ML system (the so-called "problem/input" and "feature" spaces [49]). In some circumstances [50], the attacker cannot interact with the ML model, but they can obtain a "surrogate" ML model—which is used, e.g., to *transfer* adversarial examples [44].

STRATEGY. To reach their GOAL, attackers exploit their KNOWLEDGE and CAPABILITIES and enact a STRATEGY, e.g.: attackers that fully know a deep learning model can use its gradient to craft adversarial examples [51], while attackers who can manipulate training data may add a *backdoor* [52].

We observe, however, that the terminology above has a limitation: it focuses exclusively on the ML *model* (the inner, dotted box in Fig. 1), which is just a single component of the much more complex ML *system* (the outer box in Fig. 1). We will elaborate on how to overcome this limitation in §V.

Adversarial ML in practice. Several researchers have investigated whether the explosive popularity of adversarial ML in research papers has been received with equal interest from industry practitioners. Let us trace the major findings since 2020:

(2020) Kumar et al. [3] conducted a survey of 28 companies. Participants were asked about their perspectives on adversarial ML. Most participants did not know how to respond and were primarily worried about *poisoning* attacks.

(2021) The following year, Boenisch et al. [2] raised an important warning: many ML developers confessed that "I never thought about securing my ML models."

(2021) Sun et al. [1] studied ML models deployed on mobile apps and (perhaps unsurprisingly) found that: "41% of ML apps do not protect their models at all."

(2022) A year later, Bieringer et al. [4] interviewed developers of ML and concluded that "most lack adequate understanding to secure ML systems in production", even though one third "feel insecure about adversarial ML."

(2022) Grosse et al. [5] interviewed hundreds of practitioners asking their opinion on countermeasures to adversarial ML attacks. The general consensus was "Why do so?"

We find it surprising that the viewpoint of practitioners has barely changed, even in spite of warnings from sources beyond the research domain. For example, in 2020 Gartner predicted that "by 2022, 30% of cyberattacks will leverage training-data poisoning, model theft, or adversarial examples." [53]

> OBSERVATION: Despite abundant evidence showing that ML models are vulnerable, practitioners persist in treating such threats as low priority.

C. (Economics of) Cybersecurity

An elementary principle. It is well known that "there is no such thing as a foolproof system" [54] (the actual quote is from a renowned con artist [55]). Given a sufficient amount of resources, any attacker can succeed in their goal [56]. Indeed, the purpose of security mechanisms is

[2]We acknowledge that such color-based terminology may offend some people. We use it when discussing prior work to preserve fidelity to the original sources, but we propose that future work refrain from using it (§V).

to *raise the cost sustained by the attacker* [57], either by requiring more resources (e.g., time to succeed) or by increasing the consequence of failure (e.g., getting caught). However, countermeasures also have costs (e.g., implementation efforts, periodic evaluations, maintenance, and computational resources [58], [59]) that must be factored into any decision about whether to build or deploy them. Simply put, operational cybersecurity is rooted in economics [56], [60].

Cybersecurity in practice. Let us substantiate our previous claims with public statements made by cybersecurity professionals. On July 22nd, 2022, a tweet [61] by a Sophos AI employee revealed that "Adversarial examples are not the primary concern at Sophos." This tweet inspired an insightful discussion between renowned researchers and practitioners in this field. Konstantin Berlin, Head of AI at Sophos, explains his thinking about adversarial examples [62] as follows:

> **Head of Sophos AI**: "If you look at cybercrime in economical terms (as you should because it is a business) the optimization for an adversarial ex. is not the expensive part, it is the engineering part of building a tool that can create a diverse set of attacks with no obvious watermarks." (*source*: [63])

Berlin also stated that "Given the existence of an already large number of more prevalent attacks that do bypass detections, why prioritize this one?" Such a simple (and recent) use-case shows that operational decisions are dictated by economics, and many practitioners prioritize threats that are deemed to be more important for their businesses.

> OBSERVATION: Economics is the main driver of practical cybersecurity—both for attackers and defenders.

We report the rest of the discussion of such Twitter thread in Appendix A-B, where we will also provide our own viewpoint.

D. Related Work

Many papers share our goal of improving the security of real-world ML systems. The authors of [14] and [64] provide practical guidelines on ML *for* security—whereas we focus on the security *of* ML. Some related papers (e.g., [2]–[5]) focus exclusively on the industry perspective; while most literature surveys (e.g., [39], [41], [65]) focus just on the research perspective. We consider both perspectives, because we aim to bridge the existing gap between these two sides. Perhaps the closest effort to our paper is the (unpublished) work by Gilmer et al. [66], which attempts to contextualize the implications of adversarial ML in the real-world. Despite sharing our goal, [66] has two limitations: first, it mostly focuses on computer vision and deep learning—which, as we will show, represent only a subset of the conceivable applications of ML in reality; second, the remarks made in [66] relate to papers published before 2018—i.e., almost five years ago, when ML deployments were not as popular as today.

We briefly summarize two orthogonal research areas to our paper. (i) The performance of ML tends to degrade over time [67]-[68]. This phenomenon (called "concept drift") is due to changes in the data distribution that are physiological in nature and sometimes even beyond an attacker's control. (ii) ML can also be used as an attack vector (e.g., "deepfakes" [69], or "attribute inference" [70]). This research area (i.e., "offensive ML") is outside our scope, because we focus on the protection of ML systems.

To the best of our knowledge, ours is the first position paper (on ML security) whose main arguments stem from observations based on (i) a systematic analysis of recent research, and on (ii) case studies from real experience.

III. CASE STUDIES (REAL EXPERIENCE)

We now describe three case studies fostering the contribution of industry practitioners. Our intent is twofold: (i) to elucidate some practical aspects that may be obscure to researchers, and (ii) to induce practitioners to be more open with their techniques.

The first case study (§III-A) presents the architecture of a real INVISIBLE ML system deployed at Facebook, the largest online social network (OSN). The second (§III-B) describes the triaging of phishing webpages that confused a security company's ML detector. The third (§III-C) sheds light on the role of time and domain expertise in devising query-efficient attacks during an ML evasion challenge organized by renowned tech companies. In Appendix A-D, we provide a fourth case study showing that even "adversarially aware" malware detectors can easily be fooled with expert knowledge.

Because our case studies consider ML systems for *cyberthreat detection*, this section will focus on "evasion" attacks, i.e., misclassifications of malicious input at test time.

A. "The four A's of abuse fighting": ML Systems in an OSN

This case study reflects the experience of Facebook, a *real* OSN that makes ample use of ML (in various forms) to manage its services and, in particular, to fight spam abuse.

Scenario. An attacker attempts to spread spam on Facebook; for example, they want to post a pornographic image with some text, which may lure a user to click on an embedded URL. Facebook does not allow these activities on their platform, and hence employs an ML system to prevent this type of content from appearing. The attacker—aware of the existence of the ML system—tries to evade the detector by perturbing the content and/or changing their behavior.

Problem. On the surface, this scenario represents the "evading the ML spam detector" setting that is typically envisioned in research papers (e.g. [71], [72]). However, most such papers assume that the ML system consists of (i) a single ML classifier that (ii) analyzes content and (iii) outputs the decision to either block or allow the content—a decision which (iv) is readily observable by the content's author (e.g., [73]). Such a "black-box" scenario (which has also been envisioned in its "white-box" variant [74]) assumes an adversary who can send an arbitrary number of requests to the ML model and use its response to craft adversarial examples—potentially with the added constraint of preserving the original input semantics.

Reality. Production systems however, are far more complex (see §II-A). In fact, Facebook views content classification

(block/allow) as the last line of defense in an entire funnel of countermeasures that work in concert to provide defense at different layers; to be successful, an attacker must evade each layer in the funnel. We denote this funnel as "the four A's of abuse fighting": AUTOMATION, ACCESS, ACTIVITY, and APPLICATION. We provide an overview in Fig. 3 and explain each term below. Without loss of generality we can assume that the input to the ML system is an *action* (e.g., posting an image) executed by a given *entity* (e.g., user account).

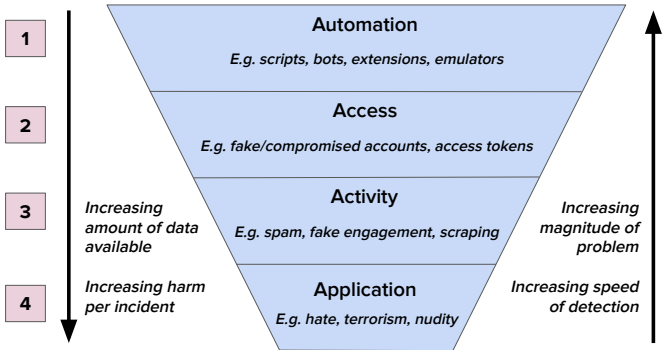


Fig. 3: Example of Facebook's ML system for spam detection. The system consists of a "funnel" of four interconnected defensive layers, each with its own logic. The attacker must bypass all layers to be successful.

1) AUTOMATION *layer.* At the top of the funnel is a simple question: "is the action due to automation?" Anti-automation is the most general defense because it applies to both read and write surfaces and to logged-in and logged-out traffic. Indeed, Facebook is constantly targeted by bots and automated scripts [75]. The AUTOMATION layer automatically prevents "black-box" attack strategies based on repeatedly querying the system over short time frames.

2) ACCESS *layer.* Automation detection is Facebook's principal defense against logged-out reads at scale. In contrast, an account is required for writes (e.g., posting content) or logged-in scraping. Most malicious activity is perpetrated by fake or compromised user accounts [76]. An attacker can, however, find other ways (e.g., fake advertiser accounts or stolen access tokens) to obtain the required permissions to launch an offense. Simply put, the role of the ACCESS layer is to analyze any write request and determine whether the corresponding write permission was obtained legitimately by the requesting entity.

3) ACTIVITY *layer.* If Facebook fails to stop a malicious action just for being automated (in the AUTOMATION layer) and does not catch the bad actor when they obtain access (in the ACCESS layer), then the next opportunity is when they begin carrying out their (malicious) activity. To reach their goal, spammers and other abusers must act in fundamentally different ways from normal users (see Appendix A-G1 for an explanation). Such differences include how they interact with the social graph [77], the distribution of the actions they take, and/or how fast they execute their actions.

4) APPLICATION *layer.* The first three layers of the funnel are generic and mostly boil down to well-known cybersecurity practices, some of which may not require ML. However, if the funnel ended at the third layer, two concerns would arise: (i) What happens if an attacker bypasses all three layers? and (ii) What about specific prohibited content such as nudity, hate speech, or gun sales—which are activities that, despite being against Facebook's policies, can be produced by "benign" users? To address these concerns, Facebook has a fourth, APPLICATION layer. Assuming that the above layers are mostly effective (i.e., low false positives/false negatives), most of the violations in this layer will be small-scale and driven by real users. This layer is where the deep learning models typically targeted in research papers [74] (e.g., nudity detectors or hate-speech classifiers) can really shine.

Complexity. Facebook's spam-detection system has numerous additional complexities not shown in Fig. 3. For example, a blocked attacker typically has no way of knowing which layer "caught" them and therefore must try many different evasion strategies. Each layer itself may contain both shallow learning and deep learning methods as well as heuristic rules, further complicating the attacker's task. The layers can inform each other even if an attacker is not blocked—for example, an account may be given a smaller rate limit threshold (ACTIVITY layer) if a fake-account detector (ACCESS layer) returns a "suspicious" score but is not confident enough to block outright. Finally, even if some layers output a decision of "malicious," such output is postprocessed by a *response* layer that decides which action to take: Ban the user? Remove the content? Show a CAPTCHA? Wait and see? The attacker may even observe different responses from multiple instances of the same input, depending on factors beyond the attacker's control—a characteristic of an INVISIBLE ML system (§II-A).

> OBSERVATION: Real ML systems include many components, not necessarily all using ML. Attackers must bypass all components to be successful.

We remark, however, that the complexity of real ML systems does not necessarily mean that such systems are "omnipotent." The following case study will reveal that even full-fledged ML detectors can be thwarted via simple tactics.

B. Are there Adversarial Examples in the Wild (Web)?

To assess the prevalence and nature of adversarial examples "in the wild" (i.e., in the real-world), we first consulted the AI Incident Database [78], containing over 1 600 reports of AI failures (as of Aug. 2022). Unfortunately, querying the database with the keywords "evasion" and "adversarial machine learning" yielded only 2 and 6 results, respectively. Apparently, only a limited number of (reported) incidents involve the types of attacks typically portrayed in adversarial ML papers. To gain further insight, we asked a leading cybersecurity company whether they encounter adversarial examples when manually triaging security incidents. This case study provides their answer.

Context. We consider the problem of *phishing webpage detection*. Similarly to the previous case study, the ML system is a commercial-grade detector, composed of diverse modules,

all with the underlying goal of "catching phish." Specifically, the ML system (which is CLOSED and INVISIBLE to the attacker[3]) consists of an ensemble of image classifiers, each responsible for a specific task (e.g., logo attribution, visual comparison, input form detection). The ML system leverages the principle of active learning [79]: For each input, the output is a "phishing confidence" score that is used (i) to make an automated dection (i.e., block/allow) and (ii) to improve the ML system by triaging to security analysts inputs for which the ML system is "uncertain." Such a setting fits our objective: most related research focuses on computer vision (§IV-A) and the ML system naturally suggests to the analysts which inputs can be linked to adversarial examples.

Method. We searched for adversarial examples in the ML system's usage logs, restricting ourselves to the 40 domains that (according to the cybersecurity company) were most commonly targeted[4] by phishing campaigns in July 2022. During this month the ML system analyzed hundreds of thousands of inputs, and among these 9 174 were flagged as "uncertain" by the ML system and required manual triage. We used this set as a starting point, and we began to review all such samples (i.e., screenshots). To make our in-depth analysis humanly feasible, as we visually inspected each sample we asked ourselves "can this be an adversarial example?" and decided to stop once we obtained a positive answer for 100 samples. We reached this number after going through 4 600 samples (half of our initial population). The discarded samples had nothing in common with adversarial examples, due to being "out of distribution" [80]—i.e., they were significantly different from any sample included in the training data of the ML system (e.g., a domain using a new logo, or shifting backgrounds—typical of, e.g., Netflix). We then used the 100 positive samples as basis for our in-depth analysis. Our aim was to determine the causes of the ML system's "uncertain" response and, in particular, if any cause could be traced back to the gradient-based techniques typical of adversarial examples. This entire process (first inspection and in-depth analysis) was conducted by two authors who worked independently and had regular meetings to resolve issues and arrive at a consensus.

Results. Our analysis suggests that attackers rely on an arsenal of relatively simple, yet often effective, strategies. We quantify the prevalence of these strategies in Table II and show their effects on real webpages ("in the wild") in Fig. 4.

TABLE II: Frequency of different evasive strategies in 100 phishing pages that were poorly analyzed by a commercial ML-driven detector.

Evasive Strategy	Count	Evasive Strategy	Count
Company name style	25	Logo stretching	11
Blurry logo	23	Multiple forms - images	10
Cropping	20	Background patterns	8
No company name	16	"Log in" obfuscation	6
No visual logo	13	Masking	3
Different visual logo	12		

[3]The attacker can only inspect the output by subscribing to the company's security product, fetching the phishing page, and waiting for the back end to recognize the webpage as phishing and eventually add it to a blocklist.

[4]We use the term "targeted" to denote a phishing webpage that is crafted so as to resemble a (benign) page of a specific website.

The most prevalent attacks include cropping, masking, stretching, and/or blurring techniques, all of which induce optical character recognition (OCR) algorithms to incorrectly extract text from the page. Such "anti-OCR" techniques have been employed for decades by phishers [81] and can (still) ably fool ML systems without relying on gradient computations. These methods typically target the *company name* when it is part of a logo, mostly by manipulating the logo's visual representation. We also found samples with a different logo altogether, as well as "company name style" attacks that alter the logo's company name. Even more rudimentary attacks simply remove the logo and/or the company name (or even both of them in 6 cases). In rarer cases, we encountered "multiple form/image" attacks, which either overlay duplicate forms on top of one another or provide a grid of forms or background images; and "background pattern" attacks, which alter the background (especially behind company logos).

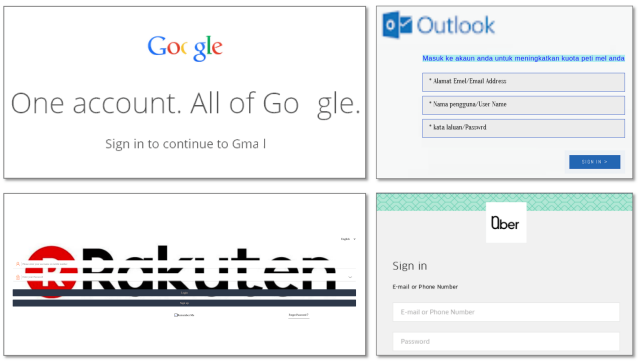


Fig. 4: Four evasive phishing samples, depicting the use of masking, cropping, blurring, and misspellings to disrupt the detection of company names, logos, and login-related keywords (e.g., "Passwrd").

Considerations. Our analysis clearly indicates that real adversaries do attempt to evade anti-phishing ML systems that use image classification, and do so with some degree of success. Although our results suggest that the strategies employed by real attackers have little in common with gradient-based adversarial examples, we cannot make such claims with certainty. Indeed, *proving* whether a given "evasive sample" is a true adversarial example or just the result of educated guessing is extremely difficult today. This problem could potentially be addressed with digital forensics techniques (which are covered by two recent works in the context of adversarial ML [82], [83]). In a sense, the only way to be 100% certain that a sample has been generated with a gradient-based strategy would be to ask the attacker directly (i.e., a "probatio diabolica"). Nonetheless, we are confident in the results of our analysis: the strategies we described above are simple yet effective. Hence, it is sensible to conclude that an attacker would opt for these methods over more computationally expensive ones.

> OBSERVATION: to evade phishing ML detectors, attackers employ tactics relying on cheap but effective methods that are unlikely to result from gradient computations.

As an additional contribution, we release (in our website [17])

the 100 evasive webpages analyzed in this case study.

C. In-Depth Analysis of an ML Evasion Competition

We now turn our attention to an "anti-phishing evasion" competition organized by industry practitioners in ML security [84], who provided us with data from the competition. Our intention is to emphasize the role of *time* and *domain expertise* in evading a ML detector. Although most contestants were security enthusiasts, the rules of this competition resembled a constrained and likely scenario simulating a real attack.

Rules. Contestants were given ten phishing webpages (as HTML) and were challenged to manipulate them in such a way as to preserve the original rendering while evading seven ML phishing detectors. These detectors are CLOSED ML systems: contestants could input any webpage to the detectors, which provided a "phishing probability" (within [0–1]) as output. Winners were determined on two criteria: the number of successful evasions (probability below 0.1) and the number of queries issued to the detectors. Contestants could send an unlimited amount of submissions (i.e., sets of manipulated webpages). A public leader board was regularly updated to reflect the current rankings, showing the number of successful evasions and queries by each contestant. The challenge started on Aug. 6, 2021 and ended on Sept. 17 (42 days in total).

Results (high-level). Four teams crafted variants of all ten phishing webpages that evaded all seven detectors, thereby achieving a perfect score of 70 points. In order of ranking, these teams made 320, 343, 608, and 9 982 queries [85]. The 1st-place (two scientists from Kaspersky) and 3rd-place (a single individual) teams published their methodologies [86], [87]. We can reasonably assume that the strategy of the 4th-place team involved some automation that resulted in thousands of queries. Using query count as a cost metric (as researchers often do, e.g., [34]), leads to the conclusion that the winning solution had the lowest cost: only 320 queries, i.e., an improvement of 47% over the 3rd-place (608 queries) and of 97% over the 4th-place (9 982 queries). However, an in-depth analysis shows that this viewpoint is quite misleading.

Results (low-level). The organizers of this competition provided us with details about the temporal distribution of the submissions made by the four top-ranked teams, whose cumulative submission history is shown in Fig. 5. We can assume that the *last* submission made by each team corresponds to the point in time in which they "finished" their attack.

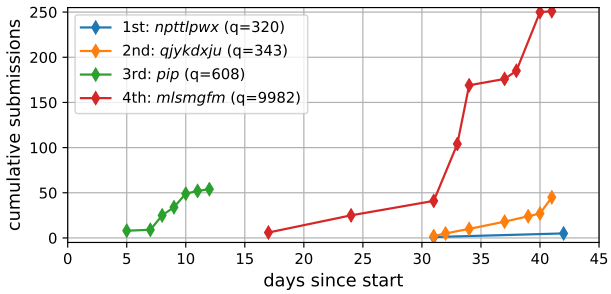


Fig. 5: Temporal distribution of cumulative submissions (y-axis) during the phishing MLSEC in 2021 (started on Aug. 6th). Each line indicates a team (q=queries). A detailed explanation of this Figure is in Appendix A-C).

The 3rd-place team (green line in Fig. 5) was the first to arrive at a perfect evasion, requiring 608 queries with the first submission on day 5 and the last on day 12. In contrast, the perfect evasion achieved by the 1st-place team (blue line in Fig. 5) was the *last* to be submitted—they made their last submission after 42 days, i.e., on the last day of the challenge. Indeed, the winning team reported adopting a "wait-and-see" approach, i.e., one that took into account the results of other participants. Originally this team wanted to use typical "black-box" techniques based on model replication (e.g., [88]), which are query-intensive. However, they changed their mind [86] when they noticed that the current leader (who ultimately finished 2nd) only had made a few hundred queries. Hence, to win they needed to be very conservative with their API calls. The winning strategy involved sophisticated manipulations of the HTML, rooted in extensive domain expertise. In contrast, the 4th-place team (red line in Fig. 5) made a large number of queries and submissions, but these were all automated and (likely) required little human effort. Moreover, the 3rd-place team required 12 days for a single person to achieve a perfect evasion, while the 1st-place team required 42 days by two people—an 8x increase in absolute time.

> OBSERVATION: Measuring attack efficiency with query counts alone does not reveal the amount of time and domain expertise required to devise a successful offense.

We have posted on our website (with permission) the code and anonymized data used in our analysis [17]).

Further Considerations. So far, we have presented an objective analysis of the evidence available on this competition. However, we believe that there is more to be learned from this case study. We provide further analysis in Appendix A-C, and in particular we attempt to derive the human effort entailed in this competition. Finally, we point the reader to our fourth case study in Appendix A-D, which further emphasizes the importance of domain expertise on both the offensive and defensive sides of an anti-malware evasion competition.

IV. SNAPSHOT OF ADVERSARIAL ML RESEARCH

We now turn our attention to the research domain. We systematically analyze all papers within our scope that have been recently published in selected "top" scientific venues. Our objective is twofold: (i) to identify research trends and (ii) to pinpoint blind spots that may inspire novel studies.

We first present an overview of our analysis (§IV-A), and then focus on some positive trends of the threat models envisioned in prior work (§IV-B for attack papers, and §IV-C for defense papers); we then highlight some inconsistent terminology adopted in research (§IV-D). Appendix B contains a detailed description of our study.

> **Disclaimer**: Our study is not a finger-pointing exercise; it is a holistic review of prior peer-reviewed work aimed at reducing the gap between researchers and practitioners.

A. Overview

Methodology. We are inspired by Arp et al. [14] and Apruzzese et al. [79]. We consider papers accepted in four venues: ACM Conference on Computer and Communications Security (CCS), USENIX Security Symposium (SEC), Network and Distributed Systems Security Symposium (NDSS), and IEEE Symposium on Security and Privacy (SP). Because we want to focus on recent trends, we consider the papers published in the last three years: 2019, 2020, 2021. We omit 2022 because (at the time of writing) it is still in progress and we want to see yearly trends. We identified 88 papers within our scope (shown in Table IV), which we then analyzed.

Main Findings. We report below the most relevant discoveries, which we will use to support some of the positions taken in this work (§V). Out of 88 papers:

- 72% focus on attacks (28% on defenses);
- 52% envision "evasion" attacks;
- 89% consider only deep learning algorithms;
- 27% do not make any mention of economics;
- 51% publicly release their source code;
- 63% carry out their evaluations on image data (only 5% consider malware, phishing, or intrusion detection);
- 20% reproduce a pipeline by building an ML model only;
- 20% experiment on real ML systems.

Positive Light: Taken *individually*, all of these papers are correct. For example, it is legitimate to perform experiments on a self-developed ML system represented by a single ML model that analyzes image data by means of DL. However, some *general* directions taken by the whole body of research generate blind spots—which we attempt to illuminate.

For additional description, interpretation, and figures showing ongoing trends we refer the reader to Appendices B-B to B-H.

B. Positive findings in Threat Models of Attack Papers

Constrained Adversaries. More and more attack papers are considering threat models that incorporate constrained adversaries. For instance, the adversary may have limited knowledge of the targeted ML system, they may have restricted query budgets, or they may be able to observe (or interact with) only a specific segment of the data processing pipeline. The underlying principle is that showing successful attacks in constrained (and thus more realistic) environments is more impactful. In particular, we highlight:

- Nasr et al. [89] consider attacks against an ML-NIDS. Specifically, they envision a "blind" adversary whose perturbations can only be applied on live network traffic, i.e., without knowledge of the packets that will be generated after those manipulated—because such packets will be generated in the (unpredictable) future. This work depicts an exemplary attack against an INVISIBLE ML system.
- Barradas et al. [90] propose FlowLens, an ML system for website fingerprinting. Here, the authors first consider attackers that are oblivious of the existence of FlowLens itself (which are unsuccesful), and then proceed to increase the knowledge/capability of the attacker. Despite it not being surprising that attackers with limited power have little success, we believe it is important to also consider cases of unsuccessful attacks (which is also done in [54]). These evaluations are useful in practice, because they portray attacks that are more likely to occur in the wild due to a lower entry barrier.

We do, however, advocate caution in assuming threat models that envision extremely constrained adversaries. Successful attacks stemming from weak adversaries (in terms of KNOWLEDGE and CAPABILITIES), but requiring sophisticated STRATEGIES, may not be very realistic—i.e., if the attacker can achieve the same GOAL with simpler STRATEGIES (see §V-B).

Unusual Strategies. In the context of evasion attacks, some papers propose STRATEGIES that have very little in common with established techniques for traditional adversarial examples (e.g., those using gradients [48]). We highlight two such papers, both of which consider attacks against ML systems for automated speech recognition (ASR).

- Chen et al. [91] aim to generate samples that trick both the ML system and humans. To do this, however, they rely on domain expertise, because prior attacks are useless against real ASR. We quote [91]: "However, even with the estimated gradients, none of the existing gradient-based white-box methods can be directly used to attack ASR."
- Zheng et al. [92] also consider adversaries who (i) know nothing about and (ii) cannot query the target ML system—which is a *real* ASR. The attack is perpetrated through educated guesses stemming from expertise in the audio domain. Indeed, we quote from [92]: "[because] the ultimate goal of ASR is [...] converting natural speech into text, we believe that the inclusion of the characteristics of natural command audios in the constructed adversarial examples may improve their transferability."

We are pleased to see some research papers that propose evasion strategies that are outside the traditional realm of adversarial ML (which typically focuses on images). Indeed, real attackers heavily rely on their domain expertise (§III-B).

Broader Goals. Some papers envision threat models wherein the attack succeeds despite the ML model's correct response. We share two such results targeting production-grade ML systems:

- Xiao et al. [22] "evade" an ML model for object recognition by eliciting an incorrect response of the preprocessing component of the broader ML system. What is intriguing is that the ML model makes the correct prediction (i.e., it outputs the correct label) for the sample received as input—but only because the ground truth of the input sample was changed after preprocessing. This example demonstrates how an attacker can succeed by fooling the ML *system* but not the ML *model*.
- Nassi et al. [25] target the ML-based object detector embedded in autonomous cars. Here, the authors (correctly) guess that such systems are often tuned to prioritize silhouettes of humans (crucial for decision-making in autonomous driving), meaning that these systems can

be tricked via "phantom" figures with no spatial depth. The authors prove their hypotheses by showing that real autonomous cars stop when a "phantom" appears. In this case, the entire ML system makes the correct response, but the attacks are successful due to specific policies adopted by the system's developers.

Simply put, attackers' goals need not require fooling the ML model or even the entire ML system. We are encouraged to see research that is exploring these scenarios.

> OBSERVATION: Assuming constrained adversaries is common. Increasingly more attacks leverage domain expertise. In some cases, attackers' goals go beyond merely causing ML models to misclassify samples.

C. Positive Findings in Threat Models of Defense Papers

Only 24 papers (out of 88) consider defenses. Compared with attack papers, defense papers tend to put a stronger focus on the *cost* of the countermeasure, typically by measuring its overhead (i.e., baseline performance degradation).

Defending against knowledgeable attackers. A common approach in defensive papers is to consider adversaries with perfect knowledge (e.g., [93], [94]). Such "white-box" attacks can be difficult to stage in reality (as also hinted in §IV-B) because acquiring complete knowledge of the targeted ML system can be expensive. In some cases, however, such knowledge can be easily acquired, especially if the targeted ML system is OPEN (see, e.g., the attacks by Sato et al. [28]). Indeed, creating an OPEN ML system is increasingly dangerous, because doing so exposes the system to attacks at different layers. For instance, Bagdasaryan and Shmatikov [95] show how to "poison" an ML system by manipulating its source code, i.e., without any change to the training data.

Defenses against limited-knowledge attackers. Tang et al. [96] propose a defense that considers (and is evaluated against) attackers who are powerful, but not omniscient; a similar "black-box" threat model is adopted also in SIGL [97]. Some (e.g., [13], [14]) may argue that similar defenses violate the "Kerchoff principle" [98], but we believe that the line of research taken by these works ([96], [97]) is worthy of being pursued due to its high value for real-world deployments of ML. Indeed, attackers without perfect knowledge are a likely threat in the wild, and hence corresponding defenses also demand attention in research. Notably, some works have shown that attackers with limited knowledge may be as dangerous as omniscient ones: Song and Mittal [99] show that "the gap between white-box attack accuracy and black-box attack accuracy is much smaller than previous estimates [100]."

> OBSERVATION: Most defenses assume attackers with perfect knowledge by default. Some, however, are tailored for attackers with limited knowledge.

We further discuss and motivate our stance on defenses tailored for limited-knowledge attackers in Appendix A-G2.

D. Inconsistencies that May Harm Future Research

During our analysis, we focused our attention on the definitions of the threat models envisioned in each paper. We report some of the confusion we encountered, with a focus on the KNOWLEDGE and CAPABILITIES of the envisioned adversaries.

What does the attacker know? The (very common) terms "white-box" and "black-box" are used in different works to denote significantly different degrees of attacker KNOWLEDGE. We highlight such inconsistencies by reporting (verbatim, emphasis ours) the descriptions provided in some papers, all of which focus on "evasion" attacks (for consistency).

- According to Co et al. [101]: "In white-box settings, the adversary has complete knowledge of the model architecture, parameters, and *training data*.[...] In a black-box setting, the adversary has no knowledge of the target model and no access to *surrogate datasets*." This description aligns with the one by Srndic and Laskov [43].
- Shan et al. [102], however, consider a different "white-box" setting: "We assume a basic white box threat model, where adversaries have direct access to the the ML model, its architecture, and its internal parameter values [...] but *do not have access to the training data*."
- Xiao et al. [22] do not specify anything about the training data: "In this paper, we focus on the white-box adversarial attack, which means we need to access the target model (including its structure and parameters)." As a matter of fact, Xiao et al. [22] also consider "black-box" attacks, which target an unknown ML model but which is trained on the exact same dataset as the "white-box" scenario.
- Suya et al. [103] assume a "black-box" attacker that "does not have direct access to the target model or knowledge of its parameters," but that "has access to pre-trained local models for the same task as the target model" which could be "directly available or produced from *access to similar training data.*" Such a definition is in stark contrast to the "black-box" one by Co et al. [101].

We also report the definition of the "gray-box" setting of Hui et al. [104] which "gives full knowledge to the adversary in terms of the model details. Specifically, except for the training data, the adversary knows almost everything about the model, such as the architecture and the hyper-parameters used for training. Note that the adversary cannot know the training data (called a whitebox)." This definition aligns with that of Shan et al. [102]—for a "white-box" threat model!

What is the "box"? Whenever "box"-based terminology is used, it is crucial to establish what is actually denoted by the "box." Recall that in the real world, attackers interact with ML systems (i.e., the dotted rectangle in Fig. 1) and not with ML models. Hence, a "white-box" attacker (resp. "black-box") should have complete (resp. zero) knowledge of the entire *system*. This consideration is overlooked in research papers: for instance, Zhao et al. [105] aim to attack "real world object detectors" and propose "white-box"/"black-box" attacks—but only consider the perspective of the single ML model. A similar case can be made for the descriptions by Demontis

et al. [44]: here the attackers' goal is first presented in terms of "*system* operation," but ten lines later the knowledge is presented with a "white-box" terminology defined as follows: "the attacker has full knowledge of the target *classifier*." Additional confusion stems from the so-called "no-box" attacks envisioned by Abdullah et al. [106], whose definition conflicts with the original one by Li et al. [50]. Specifically, the "no-box" attacker in Abdullah et al. [106] knows *nothing*, which is the de facto assumption of "black-box" attackers. In contrast, Li et al. [50] shift the focus from the KNOWLEDGE of the attacker to their CAPABILITIES: they cannot interact with the box (i.e., the ML model), but know that it behaves similarly to a surrogate ML model developed by the attacker, who also knows a subset of its training data.

What can the attacker *do*? The attacker's CAPABILITIES were also difficult to identify. In some cases, this confusion resulted from imprecise usage of the term "access." For example, Shan et al. [102] state that their attacker must have "direct access to the ML model" but do not specify whether such access includes *read* or *write* privileges; Suya et al. [103] state that the ("black-box") attacker has "no access to the target model," but has "API access to the target model" (both statements are in the same sentence). Another issue we encountered concerns the format of the *output* received by a given attacker in query-based strategies. Output can come in many forms [107], but most papers merely state that the output is "a probability" without providing additional details.

OBSERVATION: Definitions of the KNOWLEDGE and CAPABILITIES of the envisioned attackers are inconsistent, especially when using "box"-based terminology.

V. RECOMMENDATIONS

Insofar, we have made a number of observations, stemming from our preliminary discussion (§II), our real-world case studies (§III), and our detailed analysis of recent research (§IV). We summarize these observations in Table III.

TABLE III: List of original OBSERVATIONS made in our paper.

#	OBSERVATION	Ref.
1	ML models are only one component of ML systems.	§II-A
2	Academia and industry perceive adversarial ML differently.	§II-B
3	Economics is the main driver of practical cybersecurity.	§II-C
4	Evasion is achieved by bypassing all layers of an ML system.	§III-A
5	Evidence of adversarial examples in the wild is scarce.	§III-B
6	Queries are not always an effective measure of attack cost.	§III-C
7	Attackers use domain expertise and have broad goals.	§IV-B
8	Defenses can envision either strong or weak attackers.	§IV-C
9	Terminology is often imprecise and/or inconsistent.	§IV-D
10	Evading some ML systems can be very simple.	App.A-D

We can now link all these contributions and use them to support four actionable positions: adapting threat models to ML systems (§V-A), integrating threat models with cost-driven assessments (§V-B), encouraging industry and academia to collaborate (§V-C), and embracing a "just culture" for reproducible research (§V-D).

Disclaimer: We phrase all our positions with "must" because this is what *we* advocate to bridge the gap between research and practice. However, not embracing any of our positions does not invalidate future contributions.

A. Adapting Threat Models to ML Systems

Our first position is motivated by the (unintended) inaccuracies shown in some papers (§IV-D), as well as by most papers tunnel-visioning on a single ML model (§IV-A) and overlooking the much more complex ML systems (§III-A). Note, however, that complexity does not imply robustness to evasion (as shown in our case study in Appendix A-D).

POSITION: Threat models must **precisely** define the viewpoint of the attacker on **every component** of the ML system.

Holistic vision. The elements typically used when describing ML threat models (i.e., GOAL, KNOWLEDGE, CAPABILITY, STRATEGY) must be extended to cover the entire ML system.

GOAL+. Real attackers have broader GOALS than just attacking a single ML model. For example, they may want to cause damage to all tenants of an ML system [10], or simultaneously fool humans and ML classifiers [108], [109].

KNOWLEDGE+. To better portray real threats, KNOWLEDGE assumptions should cover all components of ML systems, e.g., any preprocessing steps [49]. For instance, an attacker who has *perfect knowledge* of an ML model but *zero knowledge* of the preprocessing will have *limited knowledge* of the ML system. (We provide our viewpoint on the interplay between KNOWLEDGE and domain expertise in Appendix A-G3.)

CAPABILITIES+. As with KNOWLEDGE, attacker CAPABILITIES must consider all components of the ML system. For example, attackers should only be assumed to have the ability to give arbitrary inputs to the ML model (e.g., irrespective of preprocessing) if doing so is possible in reality (especially if operating only in the feature space [42]). Moreover, the *output* of the ML system must be explicitly defined (e.g., label-only, label-and-score, top-k-scores, all-scores, logits [107]). Finally, manipulations of the training dataset should also consider the components of the ML system that collect such data.

STRATEGY+. By extending their vision to the whole ML system, an attacker can adopt STRATEGIES that "ignore" the specific ML models. For example, Hong et al. [110] apply manipulations at the hardware layer, Bagdasaryan et al. [95] manipulate the source code used to develop the ML model, and Batina et al. [111] derive additional information by spying on electromagnetic side-channels. Notably, Rakin et al. [112] describe their threat model (in terms of KNOWLEDGE and CAPABILITIES) both from the system and the ML perspectives—a promising first step towards more holistic threat models.

Precise terminology. As the foundation of every security assessment, a threat model must be unambiguous—especially in research papers. A poorly defined threat model is detrimental because it can lead to wrong estimations of attack power (e.g., the attacker may appear weaker), as well as unfair comparisons by future work. In particular, we identify four terms

that are subject to misinterpretation from either a researcher-to-researcher or a researcher-to-practitioner viewpoint. These terms are: BOX, ACCESS, EVASION, ADVERSARIAL.

BOX. We have already highlighted inconsistencies revolving around "box"-based terminology (§IV-D). Our stance is that researchers should refrain from using this terminology. Instead, we recommend (i) using *perfect*, *limited*, and/or *zero* to define the attacker's KNOWLEDGE, and (ii) precisely specifying the components of the system to which the terms apply.

ACCESS. We have already discussed some issues with this term (§IV-D). Our stance is that researchers should (i) be precise when categorizing access (e.g., *read* or *write*), avoiding ambiguous terms such as "direct" or "internal," and (ii) specify the type of access for *every* component of the ML system.[5]

EVASION. In related literature, this term denotes attacks which (i) cause misclassifications and which (ii) occur "at test time." In contrast, the generic definition of "evasion" in cybersecurity is to *bypass a detection mechanism* [113], i.e., to cause a malicious sample to be classified as benign.[6] For instance, backdoor attacks [52] are considered to be "poisoning" in the literature [39], not "evasion." However, from a security standpoint, a (real) attacker does not want to "poison" a dataset—rather, they may use "poisoning" (e.g., backdoors) to achieve their goal of evading the ML system. Our stance is that "evasion" should be used exclusively to denote the attacker's GOAL, which can be achieved via many strategies (e.g., poisoning [52], exploiting adversarial examples [114], or by ignoring the ML model completely [110]).

ADVERSARIAL. In research, it is typical to link the term "adversarial" with the ML context—most likely due to the popularity of "adversarial examples." In many conversations with practitioners, however, we found that this term generates a lot of confusion: in a security context, "everything is adversarial." Our stance is to use this term only when referring to established notions,[7] such as "adversarial example" or "adversarial ML." Nonetheless, we make an important remark, inspired by [64]: *adversarial examples are not malicious per-se*; what is malicious is crafting and using an adversarial example. Studying adversarial examples can have uses orthogonal to security, such as improving robustness of ML methods [115].

B. Cost-driven Security Assessments

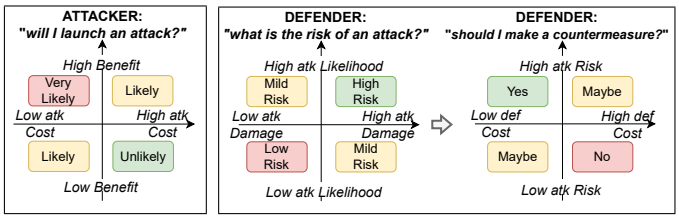
Our second statement is motivated by the poor consideration given by recent papers (see §IV-A and Appendix B-E) to the economics of cybersecurity: 27% do not mention "cost" at all.

> POSITION: Threat models must include **cost-driven** assessments for both the attacker and the defender.

Context. Biggio and Roli [39] point out that the best way to approach cybersecurity is through proactive defense: the system designer should first model the adversary, simulate the attack, and then develop a countermeasure "if the attack has a relevant impact." We stress, however, that all of these steps should also account for the cost to both the attacker and the defender. For instance, developing a countermeasure just because an attack has "high impact" may not be wise if the corresponding attack is unlikely to happen in the first place. System designers, when conducting their security assessments, prioritize threats that are more likely to occur in the wild (see §III-B). Real attackers, as pointed out by Wilson et al. [60], operate with a cost/benefit mindset: they will only attempt to evade a detector if they perceive the benefits to outweigh the costs. At the same time, no defense is foolproof [54].

> **Disclaimer**: From a *research* perspective, it is appropriate to assume (and evaluate) attackers with perfect knowledge who break any ML system. From a *practical* perspective, however, security evaluations are costly, and it is reasonable that real companies prioritize more common threats [116].

Proposal. To promote future research that has a greater impact *in practice*, we extend the recommendations of Biggio and Roli [39] with respect to *cost-driven* threat modeling. We provide an overview in Fig. 9. When establishing a given threat model, the system designer (i.e., the researcher) weighs the potential benefit to the attacker and the corresponding cost to reap such benefit (Fig. 6a). This preliminary analysis yields the likelihood of the attack [116]. The designer can then simulate the attack and gauge the corresponding damage, yielding the *risk* [117]–[119] of the attack (Fig. 6b, left). Finally, the designer can conceive of any given countermeasure, estimate its costs (e.g., implementation, overhead, upkeep), and—depending on the risk of the corresponding attack—assess whether each countermeasure should be deployed or not (Fig. 6b, right). Simply put, quoting from [64], the driver of cybersecurity is "Paying x (for the defense) to avoid paying y (if the attack succeeds), with $y \gg x$."



(a) Attacker perspective: *very likely* threats yield *high benefit* with *low cost*.

(b) Defender perspective. The attack's *risk* is assessed via its *likelihood* and its *potential damage*. On the basis of such risk, a *countermeasure* is developed if its *cost* is acceptable.

Fig. 6: Cost-driven threat modeling. The proactive security lifecycle [39] should include economic considerations for both the attacker and the defender.

We make three remarks on our cost-driven threat modeling:

- Although 57% of papers attempt to measure the "cost" of an attack/defense, they often overlook the human cost factor (e.g., time and expertise (§III-C)). Hence we encourage future studies to *account for human effort*. A possible (albeit imperfect) way of doing so is by computing the time needed to write the source code for a given

[5] An attacker can, for example, have *query* access to the ML system, meaning that they can only send an input and observe the output of the entire system; or the attacker can have *write* access to only the training data, implying no access to any other component of the ML system.

[6] In statistics terminology, an evasion corresponds to a "type II error."

[7] "Adversarial attack" and "adversarial threat model" are tautologies and can be replaced with "attack against ML" and "threat model" (Appendix A-G4).

attack/defense [120].[8] This computation can be facilitated by, e.g., using GIT repository's commit history [123], [124] or by process-mining techniques [125].

- It is wrong to consider the attacker/defender battle as a zero-sum game in economical terms. Indeed, in many cases, the gain of a (successful) attacker may not correspond to an equal loss by a (broken) defender. We discuss a (toy) use-case in Appendix A-E that makes use of the fact that cybersecurity is typically outsourced [126].
- Researchers proposing "novel" attacks should assume the viewpoint of a real adversary. An attack will not be launched if—despite initial success—the attack can be defused via simple heuristics.

Finally, a precise and holistic threat model (§V-A) is greatly beneficial for cost-driven assessments *in research*. Future work can use such descriptions to estimate the (cumulative) cost of the attack at a later point in time, and then use such an estimate as reference for comparison (even if the threat model envisions a nation-state adversary—see Appendix A-G5).

C. Collaborations between Industry and Academia

Our third position stems from the facts that (i) only 20% of our analyzed papers (§IV-A) consider real ML systems, and (ii) 90% of papers only consider deep learning algorithms, which are not necessarily those used in practice (e.g., owing to a long history of tabular data stored in production databases).

> POSITION: Practitioners and academics must **actively collaborate** to pursue their common goal of improving the security of ML systems.

The gap between research and practice would diminish if researchers had easier access to production-grade ML systems. Indeed, most real ML systems are INVISIBLE or CLOSED, preventing researchers from truly understanding how they work (as also mentioned in the Sophos Twitter thread—see [61] and Appendix A-B). For example, a researcher can reasonably assume that an OSN uses an (INVISIBLE) ML system for spam detection (e.g., §III-A); however, without knowing its internal architecture, they could hardly develop a "realistic" ML system with equivalent functionality. Similar issues exist also for CLOSED ML systems. Consider the attack by Pajola et al. [32], in which specific character modifications induce Google Translate to output a wrong translation. From the researcher's viewpoint, it is impossible to determine precisely *why* the attack was successful. (E.g., was it due to a bug in the preprocessing phase or in the ML model itself?)

We acknowledge that asking tech companies to publicly release their own ML systems is unrealistic (although some do, e.g., [127]). However, companies can still make it *easier* for academics to do security assessments. For instance, a significant barrier faced by researchers is *getting in contact* with practitioners. A potential compromise could be to offer "bug-bounty" programs that clearly define guidelines for interaction between researchers and platforms (e.g., [128]), and/or dedicating resources to processing researcher requests (e.g., [129]); in both cases companies benefit by gaining thorough, independent security assessments of their ML systems. Another possibility would be providing high-level schematics of ML systems (as in §III-A), inspiring researchers to run experiments on more realistic ML pipelines.

[8] An intriguing question is: are defenses with low computational overhead *in research* (e.g., [121], [122]) also cheap to deploy *in practice*?

D. Just Culture and Reproducible Research

Our last position stems from the observation that only half of the papers we analyzed release their source code (§IV-A). We acknowledge that there may be legitimate reasons to avoid complete disclosure (see Appendix A-F), but we conjecture that this low number is in part due to fear of being criticized.

> POSITION: In ML security, source-code disclosure must be promoted with a **just culture** [130].

A "just culture" [130] denotes an approach in which mistakes are assumed to occur and derive from organizational issues. Mistakes are avoided by understanding their root causes and using them as constructive learning experiences. This definition is in contrast to a *blame culture* [131], which seeks to avoid mistakes by actively blaming the perpetrators, with the intention of discouraging individuals from making mistakes out of fear of reputational damage. We believe that our community should avoid the latter and embrace the former.

A just culture encourages source-code disclosure, enabling the gradual improvement that is the foundation of research. In the ML security context, some attacks (or defenses) may be incorrectly implemented or evaluated, as was the case with DeepSec [132], whose evaluation was found to be flawed *after* its publication [133]. Some of these flaws, however, were spotted *because* the implementation of [132] was publicly accessible. Fear of criticism could have induced its authors [132] to not disclose the details of their platform—but *they rightly released their code*, and our community learned from such mistakes, **turning a negative result into a positive outcome**. (We are glad that [132] is still included in SP19's proceedings.)

Although we endorse future researchers to apply as much rigor as possible when performing their evaluations (i.e., before submitting their papers), we believe that authors should not be afraid of publicly releasing their source code due to the risk of others discovering flaws. Indeed, *there is much to learn from flawed implementations*: the ML domain constantly evolves, and errors can be systematized into practical guidelines (e.g., [134]) to help future research avoid mistakes.

VI. CONCLUSIONS

As our positions are adopted, we hope to see (i) more reproducible research that (ii) describes threat models for entire ML systems using precise terminology, (iii) makes cost-driven assessments factoring in human effort, and (iv) fosters active collaboration with practitioners from industry. As a final remark, we also endorse the development of novel techniques for the forensics of ML incidents: perhaps real attackers *do* compute gradients—but we cannot prove it yet!

ACKNOWLEDGEMENTS. The authors thank all participants of the Dagstuhl Seminar "Security of Machine Learning" [135], as most of the positions described in this paper derive from discussions originated during this event. The authors also thank the anonymous reviewers for their insightful feedback and valuable discussions, and the Hilti Corporation for funding.

REFERENCES

[1] Z. Sun, R. Sun, L. Lu, and A. Mislove, "Mind your weight(s): A large-scale study on insufficient machine learning model protection in mobile apps," in *USENIX Security Symposium*, 2021.

[2] F. Boenisch, V. Battis, N. Buchmann, and M. Poikela, ""I Never Thought About Securing My Machine Learning Systems": A Study of Security and Privacy Awareness of Machine Learning Practitioners," in *ACM Mensch und Computer*, 2021.

[3] R. S. S. Kumar, M. Nyström, J. Lambert, A. Marshall, M. Goertzel, A. Comissoneru, M. Swann, and S. Xia, "Adversarial machine learning-industry perspectives," in *IEEE Secur. Privacy Workshops*, 2020.

[4] L. Bieringer, K. Grosse, M. Backes, B. Biggio, and K. Krombholz, "Industrial practitioners' mental models of adversarial machine learning," in *Symp. Usable Privacy Secur.*, 2022.

[5] K. Grosse, L. Bieringer, T. R. Besold, B. Biggio, and K. Krombholz, ""Why do so?"–A Practical Perspective on Machine Learning Security," in *Int. Conf. Machin. Learn.: New Frontiers of Adversarial Machine Learning*, 2022.

[6] "MITRE ATLAS," https://atlas.mitre.org/, accessed: Aug. 2022.

[7] L. N. Tidjon and F. Khomh, "Threat assessment in machine learning based systems," *arXiv:2207.00091*, 2022.

[8] H. Anderson, "The practical divide between adversarial ML research and security practice: A red team perspective," *USENIX Enigma*, 2021.

[9] S. McGregor, "Preventing repeated real world AI failures by cataloging incidents: The AI incident database," in *AAAI Conference on Artificial Intelligence*, 2021.

[10] G. Apruzzese, R. Vladimirov, A. Tastemirova, and P. Laskov, "Wild Networks: Exposure of 5G Network Infrastructures to Adversarial Examples," *IEEE T. on Network and Service Management*, 2022.

[11] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, 1975.

[12] F. Tramèr, P. Dupré, G. Rusak, G. Pellegrino, and D. Boneh, "Adversarial: Perceptual ad blocking meets adversarial machine learning," in *ACM Conference on Computer and Communications Security*, 2019.

[13] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin, "On evaluating adversarial robustness," *arXiv:1902.06705*, 2019.

[14] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and Don'ts of Machine Learning in Computer Security," in *USENIX Security Symposium*, 2022.

[15] N. Carlini, "A complete list of all (arXiv) adversarial example papers." https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html, Aug 2022.

[16] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symposium on Security and Privacy*, 2017.

[17] "real gradients" (website with content related to this paper). https://real-gradients.github.io.

[18] M. I. Jordan and T. M. Mitchell, "Machine Learning: Trends, Perspectives, and Prospects," *Science*, 2015.

[19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, 2015.

[20] R. J. Joyce, E. Raff, and C. Nicholas, "A Framework for Cluster and Classifier Evaluation in the Absence of Reference Labels," in *ACM Workshop Artif. Intell. Secur.*, 2021.

[21] G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, and M. Marchetti, "On the effectiveness of machine and deep learning for cybersecurity," in *Proc. IEEE Int. Conf. Cyber Conflicts*, May 2018, pp. 371–390.

[22] Q. Xiao, Y. Chen, C. Shen, Y. Chen, and K. Li, "Seeing is not believing: Camouflage attacks on image scaling algorithms," in *USENIX Security Symposium*, 2019.

[23] C. Song and A. Raghunathan, "Information leakage in embedding models," in *ACM Conference on Computer and Communications Security*, 2020.

[24] S. Zanella-Béguelin, L. Wutschitz, S. Tople, V. Rühle, A. Paverd, O. Ohrimenko, B. Köpf, and M. Brockschmidt, "Analyzing information leakage of updates to natural language models," in *ACM Conference on Computer and Communications Security*, 2020.

[25] B. Nassi, Y. Mirsky, D. Nassi, R. Ben-Netanel, O. Drokin, and Y. Elovici, "Phantom of the adas: Securing advanced driver-assistance systems from split-second phantom attacks," in *ACM Conference on Computer and Communications Security*, 2020.

[26] J. Gardiner and S. Nagaraja, "On the security of machine learning in malware C&C detection: A survey," *ACM Comput. Surv.*, 2016.

[27] "OpenPilot," https://github.com/commaai/openpilot, Aug 2022.

[28] T. Sato, J. Shen, N. Wang, Y. Jia, X. Lin, and Q. A. Chen, "Dirty road can attack: Security of deep learning based automated lane centering under Physical-World attack," in *USENIX Security Symposium*, 2021.

[29] "GPT-2: 1.5B Release," https://openai.com/blog/gpt-2-1-5b-release/, Aug 2022.

[30] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson *et al.*, "Extracting training data from large language models," in *USENIX Security Symposium*, 2021.

[31] "Google Translate," https://translate.google.com/, Aug 2022.

[32] L. Pajola and M. Conti, "Fall of giants: How popular text-based MLaaS fall against a simple evasion attack," in *IEEE European Symposium on Security and Privacy*, 2021.

[33] "Clarifai," https://www.clarifai.com/, Aug 2022.

[34] H. Yu, K. Yang, T. Zhang, Y.-Y. Tsai, T.-Y. Ho, and Y. Jin, "Cloudleak: Large-scale deep learning models stealing through adversarial examples." in *Network and Distributed Systems Security Symposium*, 2020.

[35] M. Nasr, S. Songi, A. Thakurta, N. Papernot, and N. Carlini, "Adversary instantiation: Lower bounds for differentially private machine learning," in *IEEE Symposium on Security and Privacy*, 2021.

[36] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar, "Adversarial machine learning," in *Proc. ACM Workshop Secur. and Artif. Intell.*, Oct. 2011, pp. 43–58.

[37] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma, "Adversarial classification," in *ACM Int. Conf. Knowl. Discov. Data Mining*, 2004.

[38] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Machine Learning*, 2013.

[39] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Elsevier Pattern Recogn.*, 2018.

[40] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Proc. Europ. Conf. Mach. Learn. and Knowl. Discov. Databases*, 2013, pp. 387–402.

[41] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Sok: Security and privacy in machine learning," in *Proc. IEEE Europ. Symp. Secur. Privacy*, 2018.

[42] G. Apruzzese, M. Andreolini, L. Ferretti, M. Marchetti, and M. Colajanni, "Modeling realistic adversarial attacks against network intrusion detection systems," *ACM Digital Threats: Research and Practice*, 2021.

[43] N. Šrndić and P. Laskov, "Practical evasion of a learning-based classifier: A case study," in *IEEE Symposium on Security and Privacy*, 2014.

[44] A. Demontis, M. Melis, M. Pintor, M. Jagielski, B. Biggio, A. Oprea, C. Nita-Rotaru, and F. Roli, "Why do adversarial attacks transfer? Explaining transferability of evasion and poisoning attacks," in *USENIX Security Symposium*, 2019.

[45] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *USENIX Security Symposium*, 2016.

[46] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *IEEE Symposium on Security and Privacy*, 2017.

[47] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Int. Conf. Machin. Learning*, 2012.

[48] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2017, pp. 506–519.

[49] F. Pierazzi, F. Pendlebury, J. Cortellazzi, and L. Cavallaro, "Intriguing properties of adversarial ML attacks in the problem space," in *IEEE Symposium on Security and Privacy*, 2020.

[50] Q. Li, Y. Guo, and H. Chen, "Practical no-box adversarial attacks against DNNs," in *Neural Information Processing Systems*, 2020.

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

[52] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *ACM Conference on Computer and Communications Security*, 2019.

[53] Gartner, "Top-10 Strategic Technology Trends for 2020," https://forbes.com/sites/forbestechcouncil/2021/07/29/what-you-need-to-know-about-ai-security---even-if-your-company-isnt-using-ai-yet/, Tech. Rep., 2020.

[54] N. Carlini, "Poisoning the unlabeled dataset of semi-supervised learning," in *USENIX Security Symposium*, 2021.

[55] "How not to get scammed, according to a former con artist," https://www.vox.com/the-goods/2019/8/29/20836745/frank-abagnale-scam-me-if-you-can, 2019.

[56] T. Moore, "The economics of cybersecurity: Principles and policy options," *Elsevier Int. J. Critical Infrastructure Protection*, 2010.

[57] L. A. Gordon and M. P. Loeb, "The economics of information security investment," *ACM Transactions on Information and System Security (TISSEC)*, vol. 5, no. 4, pp. 438–457, 2002.

[58] R. Anderson and T. Moore, "The economics of information security," *Science*, vol. 314, no. 5799, pp. 610–613, 2006.

[59] M. De Shon, "Information Security Analysis as Data Fusion," in *IEEE Int. Conf. Inf. Fusion*, 2019.

[60] K. S. Wilson and M. A. Kiy, "Some fundamental cybersecurity concepts," *IEEE Access*, 2014.

[61] J. Saxe, "Why robustness to adversarial examples isn't a first-priority concern on the Sophos AI team." https://twitter.com/joshua_saxe/status/1550545466072264704/photo/1, Aug 2022.

[62] "Dr. Konstantin Berlin," https://ai.sophos.com/team/konstantin-berlin/, Aug 2022.

[63] K. Berlin, "If you look at cybercrime..." https://twitter.com/kberlin/status/1550959458171453440, Aug 2022.

[64] G. Apruzzese, P. Laskov, E. M. de Oca, W. Mallouli, L. B. Rapa, A. V. Grammatopoulos, and F. D. Franco, "The Role of Machine Learning in Cybersecurity," *ACM Digital Threats: Research and Practice*, 2022.

[65] G. Apruzzese, M. Colajanni, L. Ferretti, and M. Marchetti, "Addressing adversarial attacks against security systems based on machine learning," in *Proc. IEEE Int. Conf. Cyber Conflicts*, May 2019, pp. 1–18.

[66] J. Gilmer, R. P. Adams, I. Goodfellow, D. Andersen, and G. E. Dahl, "Motivating the rules of the game for adversarial example research," *arXiv:1807.06732*, 2018.

[67] R. Jordaney, K. Sharad, S. K. Dash, Z. Wang, D. Papini, I. Nouretdinov, and L. Cavallaro, "Transcend: Detecting concept drift in malware classification models," in *USENIX Security Symposium*, 2017.

[68] L. Yang, W. Guo, Q. Hao, A. Ciptadi, A. Ahmadzadeh, X. Xing, and G. Wang, "CADE: Detecting and explaining concept drift samples for security applications," in *USENIX Security Symposium*, 2021.

[69] D. Güera and E. J. Delp, "Deepfake video detection using recurrent neural networks," in *IEEE international conference on advanced video and signal based surveillance*, 2018.

[70] N. Z. Gong and B. Liu, "You are who you know and how you behave: Attribute inference attacks via users' social friends and behaviors," in *USENIX Security Symposium*, 2016.

[71] N. H. Imam and V. G. Vassilakis, "A survey of attacks against Twitter spam detectors in an adversarial environment," *Robotics*, 2019.

[72] S. Rao, A. K. Verma, and T. Bhatia, "A review on social spam detection: challenges, open issues, and future directions," *Expert Systems with Applications*, vol. 186, p. 115742, 2021.

[73] S. Madisetty and M. S. Desarkar, "A neural network-based ensemble approach for spam detection in Twitter," *IEEE Transactions on Computational Social Systems*, vol. 5, no. 4, pp. 973–984, 2018.

[74] E. Grolman, H. Binyamini, A. Shabtai, Y. Elovici, I. Morikawa, and T. Shimizu, "HateVersarial: Adversarial attack against hate speech detection algorithms on Twitter," in *ACM Conference on User Modeling, Adaptation and Personalization*, 2022.

[75] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, "The rise of social bots," *Communications of the ACM*, 2016.

[76] C. Xiao, D. M. Freeman, and T. Hwa, "Detecting clusters of fake accounts in online social networks," in *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*, 2015, pp. 91–101.

[77] T. Xu, G. Goossen, H. K. Cevahir, S. Khodeir, Y. Jin, F. Li, S. Shan, S. Patel, D. Freeman, and P. Pearce, "Deep entity classification: Abusive account detection for online social networks," in *USENIX Security Symposium*, 2021.

[78] "AI incident database," https://incidentdatabase.ai/, August, 2022.

[79] G. Apruzzese, A. Tastemirova, and P. Laskov, "SoK: The Impact of Unlabelled Data in Cyberthreat Detection," in *IEEE Europ. Symp. Secur. Privacy*, 2022.

[80] J. P. Miller, R. Taori, A. Raghunathan, S. Sagawa, P. W. Koh, V. Shankar, P. Liang, Y. Carmon, and L. Schmidt, "Accuracy on the line: on the strong correlation between out-of-distribution and in-distribution generalization," in *Int. Conf. Machine Learning*, 2021.

[81] B. Biggio, G. Fumera, I. Pillai, and F. Roli, "A survey and experimental evaluation of image spam filtering techniques," *Pattern recognition letters*, vol. 32, no. 10, pp. 1436–1446, 2011.

[82] S. Shan, A. N. Bhagoji, H. Zheng, and B. Y. Zhao, "Poison forensics: Traceback of data poisoning attacks in neural networks," in *USENIX Security Symposium*, 2022.

[83] S. Shan, W. Ding, E. Wenger, H. Zheng, and B. Y. Zhao, "Post-breach recovery: Protection against white-box adversarial examples for leaked DNN models," *ACM Conference on Computer and Communications Security*, 2022.

[84] CujoAI, "Machine learning security evasion competition (MLSEC)," https://cujo.com/machine-learning-security-evasion-competition-2021-calls-for-researchers-and-practitioners/, 2021.

[85] ——, "MLSEC 2021 anti-phishing evasion track: Task, results, and winning solutions," https://cujo.com/mlsec-2021-anti-phishing/, 2021.

[86] Kaspersky, "MLSEC 2021 anti-phishing evasion track: 1st place solution description," https://gist.github.com/KLBot/eaff9a1df3402da709a38cfc83680f0e, Oct. 2021.

[87] R. Reeves, "MLSEC 2021 anti-phishing evasion track: 3rd place solution description," https://github.com/reevesrs24/mlsec2021, Oct. 2021.

[88] J. Chen, M. I. Jordan, and M. J. Wainwright, "Hopskipjumpattack: A query-efficient decision-based attack," in *IEEE Symposium on Security and Privacy*, 2020.

[89] M. Nasr, A. Bahramali, and A. Houmansadr, "Defeating DNN-based traffic analysis systems in real-time with blind adversarial perturbations," in *USENIX Security Symposium*, 2021.

[90] D. Barradas, N. Santos, L. Rodrigues, S. Signorello, F. M. Ramos, and A. Madeira, "Flowlens: Enabling efficient flow classification for ML-based network security applications," in *Network and Distributed Systems Security Symposium*, 2021.

[91] G. Chen, S. Chenb, L. Fan, X. Du, Z. Zhao, F. Song, and Y. Liu, "Who is real Bob? adversarial attacks on speaker recognition systems," in *IEEE Symposium on Security and Privacy*, 2021.

[92] B. Zheng, P. Jiang, Q. Wang, Q. Li, C. Shen, C. Wang, Y. Ge, Q. Teng, and S. Zhang, "Black-box adversarial attacks on commercial speech platforms with minimal information," in *ACM Conference on Computer and Communications Security*, 2021.

[93] S. Hussain, P. Neekhara, S. Dubnov, J. McAuley, and F. Koushanfar, "WaveGuard: Understanding and mitigating audio adversarial examples," in *USENIX Security Symposium*, 2021.

[94] T. Du, S. Ji, L. Shen, Y. Zhang, J. Li, J. Shi, C. Fang, J. Yin, R. Beyah, and T. Wang, "Cert-RNN: Towards Certifying the Robustness of Recurrent Neural Networks," in *ACM Conference on Computer and Communications Security*, 2021.

[95] E. Bagdasaryan and V. Shmatikov, "Blind backdoors in deep learning models," in *USENIX Security Symposium*, 2021.

[96] D. Tang, X. Wang, H. Tang, and K. Zhang, "Demon in the variant: Statistical analysis of DNNs for robust backdoor contamination detection," in *USENIX Security Symposium*, 2021.

[97] X. Han, X. Yu, T. Pasquier, D. Li, J. Rhee, J. Mickens, M. Seltzer, and H. Chen, "SIGL: Securing software installations through deep graph learning," in *USENIX Security Symposium*, 2021.

[98] A. Kerckhoffs, "La cryptographie militaire," *Journal des sciences militaires*, pp. 5–38, 1883.

[99] L. Song and P. Mittal, "Systematic evaluation of privacy risks of machine learning models," in *USENIX Security Symposium*, 2021.

[100] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *IEEE Symposium on Security and Privacy*, 2019.

[101] K. T. Co, L. Muñoz-González, S. de Maupeou, and E. C. Lupu, "Procedural noise adversarial examples for black-box attacks on deep

convolutional networks," in *ACM Conference on Computer and Communications Security*, 2019.
[102] S. Shan, E. Wenger, B. Wang, B. Li, H. Zheng, and B. Y. Zhao, "Gotta catch'em all: Using honeypots to catch adversarial attacks on neural networks," in *ACM Conference on Computer and Communications Security*, 2020.
[103] F. Suya, J. Chi, D. Evans, and Y. Tian, "Hybrid batch attacks: Finding black-box adversarial examples with limited queries," in *USENIX Security Symposium*, 2020.
[104] B. Hui, Y. Yang, H. Yuan, P. Burlina, N. Z. Gong, and Y. Cao, "Practical blind membership inference attack via differential comparisons," in *Network and Distributed Systems Security Symposium*, 2021.
[105] Y. Zhao, H. Zhu, R. Liang, Q. Shen, S. Zhang, and K. Chen, "Seeing isn't believing: Towards more robust adversarial attack against real world object detectors," in *ACM Conference on Computer and Communications Security*, 2019.
[106] H. Abdullah, K. Warren, V. Bindschaedler, N. Papernot, and P. Traynor, "SoK: The faults in our ASRS: An overview of attacks against automatic speech recognition and speaker identification systems," in *IEEE Symposium on Security and Privacy*, 2021.
[107] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, "High accuracy and high fidelity extraction of neural networks," in *USENIX Security Symposium*, 2020.
[108] Y. Mirsky, T. Mahler, I. Shelef, and Y. Elovici, "CT-GAN: Malicious tampering of 3d medical imagery using deep learning," in *USENIX Security Symposium*, 2019.
[109] J. Schneider and G. Apruzzese, "Concept-based adversarial attacks: Tricking humans and classifiers alike," in *2022 IEEE Security and Privacy Workshops (SPW)*, 2022.
[110] S. Hong, P. Frigo, Y. Kaya, C. Giuffrida, and T. Dumitraș, "Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks," in *USENIX Security Symposium*, 2019.
[111] L. Batina, S. Bhasin, D. Jap, and S. Picek, "CSINN: Reverse engineering of neural network architectures through electromagnetic side channel," in *USENIX Security Symposium*, 2019.
[112] A. S. Rakin, Y. Luo, X. Xu, and D. Fan, "Deep-Dup: An adversarial weight duplication attack framework to crush deep neural network in Multi-Tenant FPGA," in *USENIX Security Symposium*, 2021.
[113] M. Handley, V. Paxson, and C. Kreibich, "Network intrusion detection: Evasion, traffic normalization, and End-to-End protocol semantics," in *USENIX Security Symposium*, 2001.
[114] L. Tong, B. Li, C. Hajaj, C. Xiao, N. Zhang, and Y. Vorobeychik, "Improving robustness of ML classifiers against realizable evasion attacks using conserved features," in *USENIX Security Symposium*, 2019.
[115] M. Fischer, M. Balunovic, D. Drachsler-Cohen, T. Gehr, C. Zhang, and M. Vechev, "DL2: Training and querying neural networks with logic," in *International Conference on Machine Learning*, 2019.
[116] "AI risk management framework," NIST, Tech. Rep., August 2022.
[117] I. Lee, "Cybersecurity: Risk management framework and investment cost analysis," *Business Horizons*, vol. 64, no. 5, pp. 659–671, 2021.
[118] M. van Haastrecht, I. Sarhan, A. Shojaifar, L. Baumgartner, W. Mallouli, and M. Spruit, "A threat-based cybersecurity risk assessment approach addressing SME needs," in *The 16th International Conference on Availability, Reliability and Security*, 2021, pp. 1–12.
[119] D. W. Woods and R. Böhme, "SoK: Quantifying cyber risk," in *IEEE Symposium on Security and Privacy*, 2021.
[120] G. Robles, J. M. Gonzalez-Barahona, and J. J. Merelo, "Beyond source code: the importance of other artifacts in software development (a case study)," *Journal of Systems and Software*, 2006.
[121] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" *Advances in Neural Information Processing Systems*, 2019.
[122] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," in *International Conference on Learning Representations*, 2019.
[123] N. McDonald and S. Goggins, "Performance and participation in open source software on Github," in *CHI'13 extended abstracts on human factors in computing systems*, 2013, pp. 139–144.
[124] E. Oliveira, E. Fernandes, I. Steinmacher, M. Cristo, T. Conte, and A. Garcia, "Code and commit metrics of developer productivity: a study on team leaders perceptions," *Empirical Software Engineering*, 2020.
[125] J. Caldeira, F. B. e Abreu, J. Reis, and J. Cardoso, "Assessing software development teams' efficiency using process mining," in *International Conference on Process Mining*, 2019.
[126] N. Kshetri, "Economics of artificial intelligence in cybersecurity," *IEEE IT Professional*, 2021.
[127] (2022) Microsoft open source toolkits. https://www.microsoft.com/en-us/research/group/machine-learning-research-group/open-source/.
[128] (2022) Meta bug bounty program info. https://facebook.com/whitehat.
[129] (2022) Youtube researcher program. https://research.youtube/.
[130] S. W. Dekker, "Just culture: who gets to draw the line?" *Cognition, Technology & Work*, vol. 11, no. 3, pp. 177–185, 2009.
[131] N. Khatri, G. D. Brown, and L. L. Hicks, "From a blame culture to a just culture in health care," *Health care management review*, 2009.
[132] X. Ling, S. Ji, J. Zou, J. Wang, C. Wu, B. Li, and T. Wang, "DEEPSEC: A uniform platform for security analysis of deep learning model," in *IEEE Symposium on Security and Privacy*, 2019.
[133] N. Carlini, "A critique of the DEEPSEC platform for security analysis of deep learning models," *arXiv preprint arXiv:1905.07112*, 2019.
[134] M. Pintor, L. Demetrio, A. Sotgiu, G. Manca, A. Demontis, N. Carlini, B. Biggio, and F. Roli, "Indicators of attack failure: Debugging and improving optimization of adversarial examples," in *Advances in Neural Information Processing Systems*, 2022.
[135] "Dagstuhl seminar 22281, "security of machine learning"," https://dagstuhl.de/en/program/calendar/semhp/?semnr=22281, Jul. 2022.
[136] D. I. Urbina, J. A. Giraldo, A. A. Cardenas, N. O. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell, and H. Sandberg, "Limiting the impact of stealthy attacks on industrial control systems," in *ACM Conference on Computer and Communications Security*, 2016.
[137] A. Erba, R. Taormina, S. Galelli, M. Pogliani, M. Carminati, S. Zanero, and N. O. Tippenhauer, "Constrained concealment attacks against reconstruction-based anomaly detectors in industrial control systems," in *Proc. ACM Ann. Comp. Secur. Appl. Conf.*, 2020, p. 480–495.
[138] S. Li, A. Neupane, S. Paul, C. Song, S. V. Krishnamurthy, A. K. Roy-Chowdhury, and A. Swami, "Stealthy adversarial perturbations against real-time video classification systems." in *Network and Distributed Systems Security Symposium*, 2019.
[139] E. Quiring, L. Pirch, M. Reimsbach, D. Arp, and K. Rieck, "Against all odds: Winning the defense challenge in an evasion competition with diversification," *arXiv:2010.09569*, 2020.
[140] F. Ceschin, M. Botacin, G. Lüders, H. M. Gomes, L. Oliveira, and A. Gregio, "No need to teach new tricks to old malware: Winning an evasion challenge with XOR-based adversarial samples," in *Reversing and Offensive-oriented Trends Symposium*, 2020, pp. 13–22.
[141] E. Edelson, "The 419 scam: information warfare on the spam front and a proposal for local filtering," *Computers & Security*, 2003.
[142] H. Aghakhani, F. Gritti, F. Mecca, M. Lindorfer, S. Ortolani, D. Balzarotti, G. Vigna, and C. Kruegel, "When malware is packin' heat; limits of machine learning classifiers based on static analysis features," in *Network and Distributed Systems Security Symposium*, 2020.
[143] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.
[144] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "FLtrust: Byzantine-robust federated learning via trust bootstrapping," in *Network and Distributed Systems Security Symposium*, 2020.
[145] H. Abdullah, M. S. Rahman, W. Garcia, K. Warren, A. S. Yadav, T. Shrimpton, and P. Traynor, "Hear "no evil," see "Kenansville": Efficient and transferable black-box attacks on speech recognition and voice identification systems," in *IEEE Symposium on Security and Privacy*, 2021.
[146] A. Salem, A. Bhattacharya, M. Backes, M. Fritz, and Y. Zhang, "Updates-Leak: Data set inference and reconstruction attacks in online learning," in *USENIX Security Symposium*, 2020.
[147] S. T. Jan, Q. Hao, T. Hu, J. Pu, S. Oswal, G. Wang, and B. Viswanath, "Throwing darts in the dark? detecting bots with limited data using neural data augmentation," in *IEEE Symposium on Security and Privacy*, 2020.
[148] V. Chandrasekaran, K. Chaudhuri, I. Giacomelli, S. Jha, and S. Yan, "Exploring connections between active learning and model extraction," in *USENIX Security Symposium*, 2020.
[149] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, "Detecting AI trojans using meta neural analysis," in *IEEE Symposium on Security and Privacy*, 2021.

[150] G. Lovisotto, H. Turner, I. Sluganovic, M. Strohmeier, and I. Martinovic, "SLAP: Improving physical adversarial examples with short-lived adversarial perturbations," in *USENIX Security Symposium*, 2021.

[151] J. Li, T. Du, S. Ji, R. Zhang, Q. Lu, M. Yang, and T. Wang, "TextShield: Robust text classification based on multimodal embedding and neural machine translation," in *USENIX Security Symposium*, 2020.

[152] Y. Chen, S. Wang, W. Jiang, A. Cidon, and S. Jana, "Cost-Aware robust tree ensembles for security applications," in *USENIX Security Symposium*, 2021.

[153] T. Eisenhofer, L. Schönherr, J. Frank, L. Speckemeier, D. Kolossa, and T. Holz, "Dompteur: Taming audio adversarial examples," in *USENIX Security Symposium*, 2021.

[154] M. Hutson, "Artificial intelligence faces reproducibility crisis," *Science*, 2018.

[155] Z. Xi, R. Pang, S. Ji, and T. Wang, "Graph backdoor," in *USENIX Security Symposium*, 2021.

[156] X. Wu, W. Guo, H. Wei, and X. Xing, "Adversarial policy training against deep reinforcement learning," in *USENIX Security Symposium*, 2021.

[157] R. Sheatsley, B. Hoak, E. Pauley, Y. Beugin, M. J. Weisman, and P. McDaniel, "On the robustness of domain constraints," in *ACM Conference on Computer and Communications Security*, 2021.

[158] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "ML-leaks: Model and data independent membership inference attacks and defenses on machine learning models," in *Network and Distributed Systems Security Symposium*, 2019.

[159] (2022) Yolov3: Real-time object detection algorithm (guide). [Online]. Available: https://viso.ai/deep-learning/yolov3-overview/

[160] Z. Li and Y. Zhang, "Membership leakage in label-only exposures," in *ACM Conference on Computer and Communications Security*, 2021.

[161] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "Textbugger: Generating adversarial text against real-world applications," in *Network and Distributed Systems Security Symposium*, 2019.

[162] R. Pang, H. Shen, X. Zhang, S. Ji, Y. Vorobeychik, X. Luo, A. Liu, and T. Wang, "A tale of evil twins: Adversarial inputs versus poisoned models," in *ACM Conference on Computer and Communications Security*, 2020, pp. 85–99.

[163] S. Ma and Y. Liu, "Nic: Detecting adversarial samples with neural network invariant checking," in *Network and Distributed Systems Security Symposium*, 2019.

[164] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *IEEE Symposium on Security and Privacy*, 2019.

[165] E. Quiring, A. Maier, and K. Rieck, "Misleading authorship attribution of source code using adversarial learning," in *USENIX Security Symposium*, 2019.

[166] L. Song, R. Shokri, and P. Mittal, "Privacy risks of securing machine learning models against adversarial examples," in *ACM Conference on Computer and Communications Security*, 2019.

[167] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, "Memguard: Defending against black-box membership inference attacks via adversarial examples," in *ACM Conference on Computer and Communications Security*, 2019.

[168] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "Abs: Scanning neural networks for back-doors by artificial brain stimulation," in *ACM Conference on Computer and Communications Security*, 2019.

[169] T. Baluta, S. Shen, S. Shinde, K. S. Meel, and P. Saxena, "Quantitative verification of neural networks and its security applications," in *ACM Conference on Computer and Communications Security*, 2019.

[170] B. Wang and N. Z. Gong, "Attacking graph-based classification via manipulating the graph structure," in *ACM Conference on Computer and Communications Security*, 2019.

[171] Z. Yang, J. Zhang, E.-C. Chang, and Z. Liang, "Neural network inversion in adversarial setting via background knowledge alignment," in *ACM Conference on Computer and Communications Security*, 2019.

[172] R. Schuster, T. Schuster, Y. Meri, and V. Shmatikov, "Humpty dumpty: Controlling word meanings via corpus poisoning," in *IEEE Symposium on Security and Privacy*, 2020.

[173] E. Quiring, D. Klein, D. Arp, M. Johns, and K. Rieck, "Adversarial preprocessing: Understanding and preventing Image-Scaling attacks in machine learning," in *USENIX Security Symposium*, 2020.

[174] K. Leino and M. Fredrikson, "Stolen memories: Leveraging model memorization for calibrated White-Box membership inference," in *USENIX Security Symposium*, 2020.

[175] X. Zhang, N. Wang, H. Shen, S. Ji, X. Luo, and T. Wang, "Interpretable deep learning under fire," in *USENIX Security Symposium*, 2020.

[176] Z. Li, Y. Wu, J. Liu, Y. Chen, and B. Yuan, "Advpulse: Universal, synchronization-free, and targeted audio adversarial attacks via subsecond perturbations," in *ACM Conference on Computer and Communications Security*, 2020.

[177] S. Abdelnabi, K. Krombholz, and M. Fritz, "Visualphishnet: Zero-day phishing website detection by visual similarity," in *ACM Conference on Computer and Communications Security*, 2020.

[178] Y. Li, M. Li, B. Luo, Y. Tian, and Q. Xu, "Deepdyve: Dynamic verification for deep neural networks," in *ACM Conference on Computer and Communications Security*, 2020.

[179] J. Lin, L. Xu, Y. Liu, and X. Zhang, "Composite backdoor attack for deep neural network by mixing existing benign features," in *ACM Conference on Computer and Communications Security*, 2020.

[180] D. Chen, N. Yu, Y. Zhang, and M. Fritz, "GAN-leaks: A taxonomy of membership inference attacks against generative models," in *ACM Conference on Computer and Communications Security*, 2020.

[181] H. Huang, J. Mu, N. Z. Gong, Q. Li, B. Liu, and M. Xu, "Data poisoning attacks to deep learning based recommender systems," in *Network and Distributed Systems Security Symposium*, 2021.

[182] S. Abdelnabi and M. Fritz, "Adversarial watermarking transformer: Towards tracing text provenance with data hiding," in *IEEE Symposium on Security and Privacy*, 2021.

[183] X. He, J. Jia, M. Backes, N. Z. Gong, and Y. Zhang, "Stealing links from graph neural networks," in *USENIX Security Symposium*, 2021.

[184] G. Severi, J. Meyer, S. Coull, and A. Oprea, "Explanation-Guided backdoor poisoning attacks against malware classifiers," in *USENIX Security Symposium*, 2021.

[185] R. Schuster, C. Song, E. Tromer, and V. Shmatikov, "You autocomplete me: Poisoning vulnerabilities in neural code completion," in *USENIX Security Symposium*, 2021.

[186] J. R. S. Vicarte, G. Wang, and C. W. Fletcher, "Double-Cross attacks: Subverting active learning systems," in *USENIX Security Symposium*, 2021.

[187] Y. He, G. Meng, K. Chen, X. Hu, and J. He, "DRMI: A dataset reduction technology based on mutual information for black-box attacks," in *USENIX Security Symposium*, 2021.

[188] H. Jia, C. A. Choquette-Choo, V. Chandrasekaran, and N. Papernot, "Entangled watermarks as a defense against model extraction," in *USENIX Security Symposium*, 2021.

[189] Y. Zhu, Y. Cheng, H. Zhou, and Y. Lu, "Hermes attack: Steal DNN models with lossless inference accuracy," in *USENIX Security Symposium*, 2021.

[190] C. Xiang, A. N. Bhagoji, V. Sehwag, and P. Mittal, "PatchGuard: A provably robust defense against adversarial patches via small receptive fields and masking," in *USENIX Security Symposium*, 2021.

[191] Y. Lin, R. Liu, D. M. Divakaran, J. Y. Ng, Q. Z. Chan, Y. Lu, Y. Si, F. Zhang, and J. S. Dong, "Phishpedia: a hybrid deep learning based approach to visually identify phishing webpages," in *USENIX Security Symposium*, 2021.

[192] A. Azizi, I. A. Tahmid, A. Waheed, N. Mangaokar, J. Pu, M. Javed, C. K. Reddy, and B. Viswanath, "T-Miner: A generative approach to defend against trojan attacks on DNN-based text classification," in *USENIX Security Symposium*, 2021, pp. 2255–2272.

[193] J. Mu, B. Wang, Q. Li, K. Sun, M. Xu, and Z. Liu, "A hard label black-box adversarial attack against graph neural networks," in *ACM Conference on Computer and Communications Security*, 2021.

[194] A. Bahramali, M. Nasr, A. Houmansadr, D. Goeckel, and D. Towsley, "Robust adversarial attacks against DNN-based wireless communication systems," in *ACM Conference on Computer and Communications Security*, 2021.

[195] L. Li, M. Weber, X. Xu, L. Rimanic, B. Kailkhura, T. Xie, C. Zhang, and B. Li, "TSS: Transformation-specific smoothing for robustness certification," in *ACM Conference on Computer and Communications Security*, 2021.

[196] X. He and Y. Zhang, "Quantifying and mitigating privacy risks of contrastive learning," in *ACM Conference on Computer and Communications Security*, 2021.

[197] M. Chen, Z. Zhang, T. Wang, M. Backes, M. Humbert, and Y. Zhang, "When machine unlearning jeopardizes privacy," in *ACM Conference on Computer and Communications Security*, 2021.

Appendix A
Additional Case Studies and Considerations

A. Evasion via Adversarial Examples (and Perturbations)

We provide here the formal definition of an evasion attack via adversarial examples and discuss the perturbations typically used to craft an adversarial example.

Definition. Let M be a (trained) ML model taking inputs in some vector space $\mathcal{V}$ and producing predictions in some label set $\mathcal{L}$. An *adversarial example* is a tuple $(x, \varepsilon) \in \mathcal{V} \times \mathcal{V}$ with the following properties:

1) $x' = x + \varepsilon$ has the same ground truth label as x.
2) M outputs the correct label for x (i.e., if $y \in \mathcal{L}$ is the ground truth label for x, then $M(x) = y$).
3) M outputs an incorrect label for x' (i.e., if y is as in (2), then $M(x') \neq y$).

We call x the *original input*, ε the *adversarial perturbation*, and x' the *adversarial example*.

This notation describes an "evasion attack" against ML; i.e., a misclassification at test time. Given x, finding a suitable perturbation can be expressed as an optimization problem (i.e., computing ε subject to some constraints).

Consideration. In related literature, the adversarial perturbation ε used to craft an adversarial example is typically very "small" (i.e., imperceptible by humans [41]). However, as pointed out by several works [13], [64], this constraint does not hold universally. Some attackers may want adversarial examples to be *noticed* by humans [109]), while in other domains (e.g., malware [49]) humans do not inspect inputs to the ML model, meaning that perturbations can be of higher magnitude—as long as they are physically realizable [114].

B. Researchers vs Practitioners (Twitter thread in §II-C)

We document here the Twitter thread [61] mentioned in §II-C and provide our viewpoint on this discussion.

Discussion (verbatim). We consider the comments between Battista Biggio (B), a well-known researcher in adversarial ML [39], [47]; Joshua Saxe (J), chief scientist at Sophos AI; and Konstantin Berlin (K), head of Sophos AI.

J1) Why robustness to adversarial examples isn't a first-priority concern on the Sophos AI team. [image showing the 'simplistic' vision of a ML malware detector by researchers, compared with the 'complex' pipeline of a real ML malware detector.]

B1) I'd tend to agree. But still optimizing over a set of transformations in a black-box manner may help find blind spots even in more complex systems – surely lp norms are not that useful here – I am talking about combinations of more interesting transformations that abuse the PE.

K1) Given the existence of an already large number of more prevalent attacks that do bypass detections why prioritize this one?

B2) The goal of all these defenses in the end is about raising the bar for the attacker, and this work in my opinion is useful in that direction. It is important that people know that ML is not learning what we expect it should learn…

K2) If you look at cybercrime in economical terms (as you should because it is a business) the optimization for an adversarial ex. is not the expensive part, it is the engineering part of building a tool that can create a diverse set of attacks with no obvious watermarks. [follows] Think of Sophos as human adversarial attackers against your adversarial attack. My bet is that if you deployed most published ML adversarial attacks at scale Sophos would block you within a few hours in such a way that you would have to rewrite your PE generator, no ML will help.

B3) It depends. The transformations used are perfectly legit (e.g. adding sections). Btw we never thought of bypassing a real AV pipeline, we know that is more complicated. We only wanted to show that some ideas (using DL from raw bytes) are much more insecure and need improvement.

The last message (B3) was sent on July 24, 2022 at 10:06 AM, while the first message (J1) was sent on July 22, 2022 at 8:17 PM (both times are GMT+1).

Our viewpoint. This discussion underscores our observation that researchers and practitioners are solving different problems, or at least are working in different threat models. The tip-off is one specific term in K2: "if you deployed most published ML adversarial attacks *at scale*." Our first case study (§III-A) offers an example of threats that operate "at scale" being prioritized by companies, and it is not surprising that Sophos would likely do the same. However, the attackers envisioned in many adversarial ML papers (and, more generally, in many security papers [136]–[138]) tend to be subtle. Hence, both sides are making valid points, but they do not appear to agree because they are not talking about the same thing.

Despite this gap (which our paper attempts to close), we believe that B3's point is valid. Clearly (§V-C), a better cooperation between industry and academia would kick-start novel research efforts that assess the robustness of full-fledged ML systems against adversarial examples. However, there is still much to learn from the results of recent literature, despite the fact that the envisioned ML systems are often inaccurate representations of real ones.

C. Further Considerations on our third case study (§III-C)

To allow a more complete understanding of our anti-phishing evasion case study (§III-C), we provide a further description of the rules of this competition. We also provide our interpretation of the events that transpired during the challenge when taking into account the human effort (and decision making) required by each top-ranked team.

1) Queries and Submissions: We clarify Fig. 5 by explaining the difference between a "submission" (y-axis) and a "query" (legend). During the challenge, participants could either: *query* the detectors—either a single one or all of them (there were 7 detectors in total); or *submit* their (adversarial) webpages. The latter counts as a "submission" (y-axis) and automatically generates 7 queries for each webpage included in the submission. For example, a participant that submits 2 webpages automatically triggers 14 queries, because each webpage is analyzed by all 7 detectors (to determine how much points to award). The organizers of MLSEC logged each submission but not the individual queries (i.e., the single API calls). This is why the plot in Fig. 5 shows the submissions

in the y-axis. We are aware that a more fine-grained analysis would consider the history of the individual queries. However, as is evident by Fig. 5, there is a strong correlation between number of submissions and number of queries (although, technically, one could issue 10k queries and make only one submission), so we reported submissions in the y-axis.

2) Interpretation (human effort): Let us summarize the **facts** (for brevity, we denote the teams as 1st, 2nd, 3rd, 4th).

- On day 0 all teams had a common objective, to use as few queries as possible.
- On day 5, 3rd made his first submission.
- On day 12, 3rd made his last submission, thereby completing his attack and leading the ranking with 608 queries. This information was publicly available: after this moment, all teams teams knew that, to win, they had to use fewer than 608 queries.
- From day 17 to day 42, 4th made various submissions and reached a perfect evasion by making 9 982 queries.
- On day 31, both 1st and 2nd made their first submission. (This does not mean that they did nothing before day 31.)
- On day 41, 2nd made their last submission, producing a perfect evasion with 343 queries. This information was publicly available.
- On day 42, *7 hours after 2nd's last submission*, 1st made their second submission, which included 9 webpages (out of 10), all of which evaded all the ML detectors.
- Also on day 42, 1st made their last submission, producing a perfect evasion with 320 queries.
- At the end of the challenge, 1st stated [86]: "At first, we thought of attempting a classic model replication attack [...] but as we started working on the competition, we noted that the leader had already achieved the highest possible score using just 343 API calls"

Now, let us **interpret** these facts. Firstly, the statement by 1st is technically incorrect, because the first submission by 1st was on day 31, and during this time-frame (i.e., between day 31 and day 42 – but also before day 31) 1st were able to do anything (though there is no record of what they did). Furthermore, 1st knew that 3rd had been leading with 608 queries since day 12: with such knowledge, why would 1st even consider attempting a "model replication attack" on day 41? Finally, the second submission of 1st (on day 42) occurred only 7 hours after the last submission by 2nd (on day 41), and this submission included 9 webpages that all achieved perfect evasion. Hence, we find it hard to believe that 1st "started working on the competition" only after noticing that the leader had 343 calls (but of course we cannot be 100% certain).

Secondly, we observe that 2nd arrived at their perfect evasion after knowing that they needed fewer than 608 queries (the leading score, from 3rd). Hence we conjecture that without the efforts of 3rd, it is unlikely that 2nd would have reached their perfect evasion in 343 queries; 2nd knew they were constrained to at most 607 queries and hence had to be more careful. Next, 2nd made their first submission on day 31 and their last on day 41, taking at least 19 days to reach the same result that 3rd had achieved in 12 days (although 2nd required roughly half the queries). We can hence conclude that "3rd required less time than 2nd to accomplish the attack."

Finally, let us go back to 1st; specifically, we focus (again) on their public statement [86]. They stated that they wanted to do a model replication (which we can conjecture required "less effort"), but they changed their mind after seeing the score of 2nd, who themselves achieved after the efforts of 3rd.

Conclusions. By connecting all these observations, we conclude that without the effort of 3rd, the perfect evasions by 1st and 2nd (with 320 queries and 343 queries, respectively) would (likely) not have been achieved. Alternatively, had the leaderboard not been publicly visible, 1st may have opted for a model replication attack (as they themselves stated!). We do not know what 2nd would have done, but 3rd would (likely) still have used their domain expertise to reach their solution—in only 12 days, and requiring 608 queries (3rd was the first contestant to reach a perfect evasion).

From a different perspective: model replication attacks require more queries (as clearly shown by 4th place), but may require less human effort (as can be extrapolated from 1st actions and statement).

D. Case Study: Malware Competition (2020)

We extend our set of case studies with an in-depth analysis of the Malware Competition organized in the 2020 edition of MLSEC. Here, we elucidate how "competition-grade" ML malware detectors can be both hardened and evaded.

Rules. This competition had two phases, each with a specific challenge: the first focused on defense, the second on attack. Let us describe what each challenge entailed.

- (Defense) In the first phase, participants were asked to submit ML systems for malware detection. Submitted solutions had to meet some performance metrics: at most 1% false positive rate, at least 90% true positive rate, and at most 5 seconds response time—all of which were measured on an *unknown* test set. Winners were determined depending on how well their solutions performed against the attacks launched in the second phase.
- (Attack) In the second phase, participants (which could be different from those of the first phase) were given 50 portable executable (PE) malware samples and allowed to manipulate them in any way that preserved the original malicious functionality. The manipulated PE had to evade as many ML-based detectors as possible—including those accepted in the previous defense phase. These "attackers" were given API access to the detectors, which resembled CLOSED ML systems: attackers could only submit an input and observe the output (i.e., the *probability* that the sample was malicious). Winners were determined on the basis of how each of the 50 manipulated malware samples performed against *all* the considered detectors.

What did the winners do? Let us focus our attention on the methodologies adopted by the winners of each challenge.

- (Defense) Quiring et al. [139] won the defensive challenge. Their solution integrated typical approaches to

counter malware, such as signature-matching and ML-based detection based on static analysis; surprisingly, however, they also integrated a defensive layer for *anticipating* adaptive attackers querying the detector—i.e., the actions of the participants in the second challenge. The solution of [139] stopped 77% of the malware submitted during the attacker's challenge.
- (Attack) Ceschin et al. [140] won the attacker challenge by inducing all of their samples to evade *all* the considered detectors, including the winning submission by Quiring et al. [139]. Notably, the strategy adopted by [140] was *simple*: they observed that "embedding the malware payload into another binary eliminates most detection capabilities presented by the models." Indeed, to quote from [140]: "[although] there are approaches for adversarial attacks generation based on complex techniques [...] we show that it is possible to generate attacks using known, simple techniques."

OBSERVATION: Bypassing some ML systems may not require sophisticated techniques (e.g., those based on gradients) commonly adopted in research papers.

E. Example: Gains and Losses is not a zero-sum game

The gain of an attacker may not correspond to the loss of a defender. Consider, for example, a company $\mathcal{D}$ whose main business is the development of ML-based malware detectors and who sells their products to a customer $\mathcal{C}$. Now consider an attacker $\mathcal{A}$ that successfully evades one of the detectors developed by $\mathcal{D}$ and manages to inject some ransomware to the systems owned by customer $\mathcal{C}$, asking for a ransom in the amount of $\mathbb{R}$. Two scenarios can occur:

- **Pay.** If $\mathcal{C}$ pays the ransom, then $\mathcal{C}$ loses $\mathbb{R}$ while $\mathcal{A}$ gains $\mathbb{R}$. However, $\mathcal{D}$ (the developer of the 'evaded' detector) loses very little. In most cases, the contract between $\mathcal{D}$ and $\mathcal{C}$ protects $\mathcal{D}$ in case of "faults" in their systems. In contrast, if the contract determines that $\mathcal{D}$ is liable for the losses incurred by $\mathcal{C}$, then $\mathcal{D}$ will pay $\mathbb{R}$ to $\mathcal{C}$; however, in this case, $\mathcal{D}$ is typically insured by other companies, who will reimburse $\mathbb{R}$ to $\mathcal{D}$. Hence, if the ransom is paid to $\mathcal{A}$, then $\mathcal{A}$ gains $\mathbb{R}$, and $\mathcal{D}$ loses nothing (assuming that $\mathcal{C}$ regains control of their systems).
- **Recover.** If $\mathcal{C}$ does not pay the ransom, then $\mathcal{A}$ gains nothing (actually, $\mathcal{A}$ may lose something, e.g., the time spent to carry out the attack). However, $\mathcal{C}$ will incur a loss, because their systems are compromised. Two cases can follow, depending on the contract terms:
  - If $\mathcal{D}$ is liable, then they must help $\mathcal{C}$ to recover their data, and $\mathcal{D}$ themselves may either incur a loss (e.g., human labour), or not (e.g., if they are insured).
  - If $\mathcal{D}$ is not liable, then $\mathcal{D}$ may *gain* something, for example if $\mathcal{C}$ asks (and pays) $\mathcal{D}$ to assist in the recovery of their systems.

Simply put, when looking at cybersecurity from the economic (and operational) viewpoint there are many nuances that can influence the decision to deploy a countermeasure in an ML system. Such considerations further complicate the decision of whether to develop a countermeasure in the first place. The latter decision is made by the system designer, whose main goal is to benefit their bottom line and for whom customer considerations are secondary. (Of course, instilling trust and attracting future customers by having the best system is always an important consideration!)

F. Unreproducible Papers

Unavoidable truth. The ideal of a publication that releases all its source code and uses public datasets containing attacks from the real world is difficult to meet in practice, especially in cybersecurity. The most common and understandable constraint is that of user privacy: real-world data is tied to consumers or employees, and custodians of this data must respect its confidentiality. The employers of security and ML practitioners may also impose constraints with regard to releasing source code, e.g., to avoid giving attackers complete vision of an ML system or a portion thereof. When constraints of this nature do not exist, researchers and practitioners must provide both descriptive details and technical artifacts that maximize reproducibility.

What can be done. Publications that face privacy constraints and/or cannot release their source code can, however, still be valuable so long as they work within those constraints to make reproducibility a priority. To this purpose, we propose the following *actionable* principles, which can be adopted by research papers to improve their scientific value:

1) ML systems and models should be *described with sufficient clarity* that a reader can re-create the model with a high degree of accuracy after reading the paper. Features used by the model should be described in detail.
2) Practitioners should *evaluate their models on public datasets* whenever data that is sufficiently similar to the private data is available. This recommendation need not prevent additional evaluation on private datasets.
3) Papers should *describe as many of the nuances of private datasets as possible* (e.g., examples of feature values and their distributions, correlation between features, and other properties of the data) while not providing individual data points when doing so would violate user privacy.
4) *Attacks* that appear in private datasets should be *released when possible*. If they are not releasable, they should be described with sufficient detail to enable researchers to reproduce similar attacks in their own studies and to understand attacker threats that appear in real systems.

We observe, however, that providing all such details can be tough in a research paper: as highlighted in [64], some venues (especially conferences) have a strict page limit. Hence, *we endorse conference organizers to remove such limits* when they are used to provide additional technical details that are not part of the paper's contribution but which are crucial to provide at least some form of reproducibility.

G. Discussion (with the Reviewers)

For informational purposes, we record here some of the discussion we had with our paper's (anonymous) reviewers, since other readers may have similar questions. Transcripts have been lightly edited for clarity.

1) Spammers and regular users: According to the description in §III-A, the malicious user wishing to upload a problematic image seems to perform a very similar action to millions of users uploading any sort of images every day on any OSN. Hence, although some malicious user's action pattern may appear different from those of normal users (especially when automated tooling is used to perform such actions) this is far from being an universal truth.

Our Response: This statement is correct, and we will happily clarify. Ultimately, "professional spammers" are trying to make money. They may do this by, e.g., posting links to counterfeit goods; "phishing" legitimate users and selling their account credentials; engaging in 419 scams [141]; or in other ways. Since the expected value of a single spam post is very low (most people don't buy fake Ray-Bans!), spammers need to post **a lot** to reach their goal. So they necessarily need to turn to techniques—usually, but not always, automation—that allow them to distribute content quickly across a wide audience. This type of behavior does not match that of a typical Facebook user. Clearly, low-volume spammers who invest a lot of effort in mimicking the behaviour of legitimate users are harder to detect (with or without ML). However, since (a) such an attack requires a high resource investment; (b) the expected value (to the spammer) of a single spam post is still low; and (c) most (but not all) spammers are economically rational, such attacks will be rare and it's acceptable for the ACTIVITY layer to miss them. (Of course, the spammy content may still be caught by the APPLICATION layer.)

2) Defenses vs. limited-knowledge attackers (§IV-C): It is true that perfect-knowledge attacks are rare, but we should be very careful to draw a distinction between "defenses that are robust in a query-only setting" (a reasonable threat model) and "defenses that only work because the attacker doesn't know what the defender is doing" (an *un*reasonable threat model).

Our Response: This observation is legitimate. We do not advocate that "all (defensive) papers must consider attackers without perfect knowledge." Clearly, it would be beneficial if all papers proposed defenses that work against perfect-knowledge attackers. Our intention, however, is to emphasize that "real" attackers may not have perfect knowledge, and hence we find it positive that *some* papers propose defenses specifically addressed at such attackers. Put differently, we think it is unfair that papers may be rejected because "the threat model considers an attacker without perfect knowledge": if the threat is (justified to be) likely to occur in reality, then an effective defense can be beneficial to the real world. We believe there is much to learn from these evaluations as well!

3) Domain Expertise: In §III-C, the 1st–3rd place teams all reached a perfect evasion by exploiting their domain knowledge. Does assuming attackers with different degrees of domain knowledge describe different threat models?

Our Response: Tough question. In this case study, all participants had the same KNOWLEDGE, i.e., they knew *nothing* of the ML system (aside from that it analyzed webpages in the form of HTML). However, clearly, all participants had different "knowledge" of the respective domain. Note, however, that such knowledge is impossible to quantify: maybe they all perfectly knew the in-and-outs of HTML, but a participant was simply "lucky enough" to make a correct initial guess, which they then leveraged to build the remaining parts of the attack. In a sense, this case study (§III-C) shows that *guessing is a crucial part of a real attack* (and we find promising that increasingly more papers are incorporating guessing in their attacks, e.g., [10], [95]). Yet, we believe that such different degrees of "domain knowledge" should not be classified as different threat models. From a security standpoint, it is crucial to envision an attacker that is expert in the general domain. Note, however, that this is different from assuming that the attacker has "perfect KNOWLEDGE of the target ML system"! Hence, we conclude that the threat model should simply state the KNOWLEDGE of the attacker with respect to the ML system, and then assume that the attacker is an expert in any of the components of such ML system that they are aware of.

4) "Adversarial" in practice: Can you clarify what you meant by "everything is adversarial" in §V-A?

Our Response: From the perspective of a security practitioner, an event is either "benign" or "malicious"; i.e., either the system is under attack or it is not. Hence, the term "adversarial attack" is redundant: if there is an attack, then by definition there is an adversary and the attack is "adversarial." Similarly, a "threat model" implicitly assumes that there is "a threat" (i.e., an attacker) which is obviously adversarial.

In our conversations, security practitioners are often confused when we refer to "adversarial attacks." From a research perspective this term refers to "attacks that exploit the vulnerability of ML models to adversarial examples," while from a practitioner perspective the focus is just "attack." In other words: the term "adversarial attack" implicitly means that some attacks are not "adversarial" — which is illogical.

To provide examples, sometimes in our exchanges with practitioners (P), the following occurs:

- P: "Our systems are constantly under attack."
- Us: "Are the attacks adversarial?"
- P: "Well of course they are!"

Alternatively:

- Us: "Do you test your systems against adversarial attacks?"
- P: "Are there attacks that are not adversarial?"

(An even more awkward situation would arise from asking practitioners, "Are you more scared of adversarial attacks, or of non-adversarial attacks?")

We believe that a more precise (i.e., less redundant) usage of the term "adversarial" will benefit bridging the gap between research and practice in the context of...adversarial ML!

5) Nation-state adversaries: If we consider nation-state adversaries, is human cost measurable here (or relevant)?

Our Response: It depends. Generally speaking, if a "nation-state adversary" is assumed to have unlimited resources, then we agree that it makes little sense to report such a cost measure. However, such (extreme) cases should be reported in the paper when presenting the threat model. For instance, a paper should state "we assume an attacker that is backed by an entire country, and which is hence willing to invest a large amount of resources to carry out their strategy." Explicitly stating this assumption would automatically put such a paper in a different bracket with respect to attacker cost. As an aside, if such a "nation-state–sponsored attack" is found to be "devastating" (e.g., it leads the entire ML system to malfunction) then real companies would be more interested in studying the corresponding paper and devising appropriate countermeasures. (In a sense, what we have just described is a use case of "Adapting Threat Models to ML Systems" presented in §V-A). Nevertheless, in reality, even nation-state adversaries do not have unlimited resources: hence, making such an assumption by default is not very realistic.

Appendix B
State-of-the-Art: Literature Review

In this Appendix we describe in more depth our literature review summarized in §IV. Given the detailed methodology, the systematic analysis, and the original interpretations, we consider this appendix to be a complementary contribution of our position paper.

A. Methodology

Inspired by by [14] and by [79], we carry out our review by adopting a structured approach split into two phases, *selection* and *inspection*. Both phases involved exchanges of opinions among the authors of this paper, aimed at ensuring fairness in the review process and removing potential sources of bias.

1st phase: Selection. We conducted our survey in July and August 2022. During this time frame we performed two steps: we *acquired* the proceedings and *filtered* them.

i) *Acquire.* We downloaded the proceedings of the "Top 4" security conferences (SEC, NDSS, CCS, SP). We considered the editions of these conferences held in the previous three years, i.e., in 2019, 2020, and 2021. We only consider the papers published in the main event (i.e., we do not consider workshops). Overall, these proceedings contained 435 papers for 2019, 470 for 2020, and 644 for 2021.

ii) *Filter.* To identify papers within our scope, we started by considering papers included in those sections of the proceedings that specifically focused on "machine learning." We then extended our search by also performing a summary scan of the titles and abstracts of the remainder of the proceedings, identifying additional candidate papers.[9] At the end of this step, we obtained 30 papers for 2019, 35 papers for 2020, and 66 papers for 2021.

[9]E.g., the paper [142] is included in the "Malware 2" section of NDSS20, but it (also) considers attacks against ML-based malware detectors

2nd phase: Inspection. We analyzed these 131 papers, with two objectives: *filtering* those papers that, despite being related to ML security, were beyond our scope; and *distilling* the knowledge that we wanted to communicate in our work.

i) *Filter.* Our focus is on papers on the security of ML and, in particular, on attacks (and defenses) against ML systems. We have already discussed (see §II-D) some orthogonal research areas. Upon inspecting our candidate papers, however, we found 3 additional areas to exclude from our main analysis due to assumptions that significantly deviate from our main focus. These three areas are *federated learning* [143], where the corresponding issues (such as byzantine fault tolerance [144]) mostly pertain to the distributed systems domain; *robust, efficient, or secure computation*, because they mostly focus on technical implementations and do not propose or consider any adversarial ML attack; and *high-level analyses*, because they do not propose any original attack or defense in the ML security context. Eventually,[10] we obtained 23 papers for 2019, 24 papers for 2020, and 41 papers for 2021.

ii) *Distill.* Finally, we thoroughly analyzed the final set of 88 papers by going through each paper and answering a set of 12 questions. Each question focused on deriving knowledge significant for our position paper, as well as for promoting future work in this research field. Although most of our questions can be easily answered, some are not straightforward (as we will discuss). To minimize the likelihood of errors and reduce subjective bias, we carried out our analysis in pairs: two authors independently reviewed each paper and then discussed the findings in a series of meetings. We repeated this process five times.

Before we delve deeper into our analysis, we provide some interesting trivia. In 2019, most of the relevant papers were put in sessions entirely devoted to "ML security." In 2021, however, we found sessions entirely dedicated to specific attacks (e.g., "attacks on speech recognition" or "inference" at SP2021). Some papers have very similar contributions across the same conference. For instance, [91] and [145] (both at SP21) propose attacks on speech recognition (and both, surprisingly, evaluate their proposals also on real humans); whereas [146] (SEC20) and [24] (CCS20) propose membership inference attacks that leverage the updates of ML models.

B. Research Questions

We inspected 88 papers by asking ourselves 12 research questions, divided into two groups.

Generic Questions (G). The first eight (G1 to G8) are generic, and are meant to provide a broad overview of the latest trends in research—some of which are reported in our main paper (§IV). For each of the 88 works, we ask ourselves:

G1) Does the paper *focus* on an attack or on a defense?

G2) What is the main attack *family* (i.e., poisoning, stealing, evasion, membership inference)?

[10]We provide in our website [17] the list of 43 papers that we excluded.

G3) What *paradigm* of the ML model that is subject to the attacks (i.e., does it rely on shallow or deep learning)?
G4) Are the *costs* taken into account (in any way)?
G5) What are the *data-types* (i.e., images, audio, text, or other) considered in the evaluation?
G6) Has the *source-code* been publicly released?
G7) Has a complex *pipeline* been reproduced in the evaluation (i.e., does the ML system consist in just an ML model)?
G8) Does the paper consider an ML system *deployed* in the real world (and, if yes, what is its *type*)?

The answers to G1 to G8 are shown in Figs. 7 to 13, all sharing the same structure. Each figure refers to a specific question, and presents four stacked bars, one per year [2019–2021] while the rightmost bar is an aggregate. Each stacked bar reports the answers to the corresponding question, both in terms of relative (y-axis) and absolute (written on the specific bar) frequency over a given stack. Descriptions and interpretations on these results are provided in appendices B-C to B-G.

Threat-Model Questions (T). The last four (T1 to T4) relate to the threat model envisioned in the paper. Specifically, we consider the *weakest* (we explain our reason in Appendix B-I) attacker assumed in a paper, and then ask ourselves:

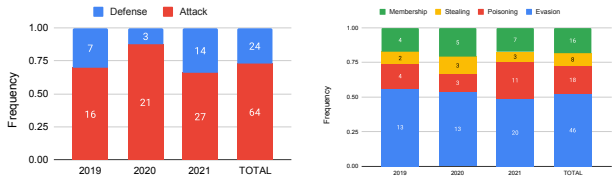
T1) Does the attacker know the ML model's *parameters*?
T2) Does the attacker know the *semantics* of the input data fed to the ML model?
T3) Can the attacker observe the *output* of the ML model (and, if yes, *how*)?
T4) Does the attacker have any power on the *training set*?

We explain how we answered T1 to T4 in Appendix B-H.

The answers to all our questions for all our considered papers are provided in Table IV, described in Appendix B-I.

C. Answers to G1 and G2: Focus, and Attack Family

The answers to the first two questions are shown in Figs. 7.



(a) G1: What is the *focus* of the paper? (b) G2: What is the main attack *family*?

Fig. 7: Answers to G1 and G2.

G1: Focus. This question is straightforward to answer: does the paper focus on attacks against ML, or on defenses to attacks against ML? In some rare cases, some papers simply proposed a ML method to solve a given task (e.g., intrusion detection [147]) but also evaluated this method against some evasion attacks: we considered these as attack papers, because the proposed ML method is not typically designed to withstand adversarial ML attacks. From Fig. 7a, we can see that 75% of papers put a stronger emphasis on the attack—although most of these papers (e.g., [54]) also discuss some countermeasures to the corresponding attack. This result is not surprising, as 'attack papers' are typically considered to have a higher novelty value—because the attack stems from a different threat model, or considers a new domain.

G2: Attack Family. Inspired by [4], we consider four main attack families: *poisoning*, *model stealing*, *evasion*, as well as *membership inference*. Fig. 7b shows that *most papers envision evasion attacks* (∼50%), whereas poisoning, model stealing, and membership inference attacks are less prominent (20%, 10% and 20%, respectfully). However, we can see a decreasing trend of evasion, and a *rising trend for poisoning attacks*, especially in 2021. We conjecture that this phenomenon is due to the 2020 paper by Kumar et al. [3], highlighting that industry is particularly worried about poisoning.

D. Answer to G3: ML paradigm

This question is also straightfoward: is the underlying algorithm used by the 'targeted' ML model based on shallow or deep learning? The results are in Fig. 8.

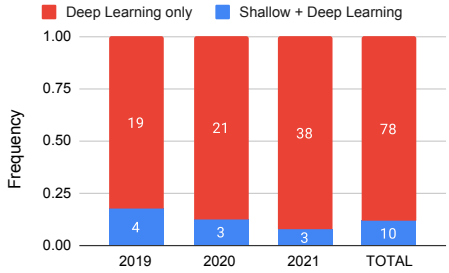


Fig. 8: G3: what is the considered ML *paradigm*?

It is apparent that most papers consider ML systems *exclusively based on DL* (i.e., those using neural networks). Only 10 papers (out of 88) evaluate ML systems based on shallow learning (SL) algorithms (interestingly, two papers consider *only* SL: [49] and [90]). We believe that future efforts should put more focus on SL, for a twofold reason:

- *in some domains, SL is better than DL* [21], [148]. E.g., we quote from [142]: "Even though in our experiments we used SVM, deep neural networks, and different variants of decision-tree learners, like random forest, we only discuss the results of the random forest approach as (1) we observed similar findings for these approaches, with random forest being the best classifier in most experiments, and (2) random forest allows for better interpretation of the results compared to neural networks."
- *SL exhibit different properties than DL.* We find insightful to quote a statement by Xu et al. [149] (emphasis ours): "we mainly detect AI Trojans on neural networks. We do not include other ML models in our discussion mainly because *there is no current research showing that they suffer from backdoor attacks*." Moreover, some SL methods (e.g., Decision Trees) do not employ gradients, meaning that some gradient-based attacks (as well as gradient-based defenses) may not work on them.

We also mention that our case study (§III-A) elucidated that SL methods are used *also* in commercial-grade ML systems.

E. Answer to G4: Cost

As we discussed (§II-C), the *cost* (i.e., the economical factor) plays a crucial role in operational cybersecurity. When analyzing prior work, we considered three possible answers to G4: whether a paper *measured* the cost in some way (e.g., performance tradeoff, required queries); whether this cost was at least *mentioned* or discussed; and whether *no consideration* on the cost was made. These answers can cover both attack and defense papers. The results are shown in Fig. 9.

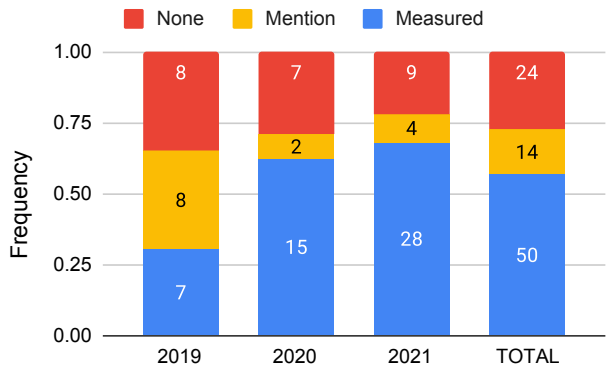


Fig. 9: G4: are the *costs* taken into account (in *any* way)?.

These results show a positive trend. In 2019, less than 30% of the papers did some measurements of the cost, and 30% made no consideration whatsoever. The prevalence changes in 2020 and 2021, as less than 25% papers did not take into account the economical factor, whereas more than 60% performed actual measurements.

Cost: Attack papers. We highlight three papers which attempted to *quantify* the cost of an attack in real currency.

- (Equipment) Lovisotto et al. [150] evade object detectors via artificial light manipulation through *commercial projectors*. Some of these projectors have an MSRP of few hundreds dollars, whereas others are more expensive.
- (Queries) The authors of [34] and [148] alongside measuring the cost of the proposed attack in terms of API calls (i.e., queries), but also quantified the actual cost required to make these API calls. Specifically, they considered the *prices of popular MLaaS*, and derived that their attacks only necessitate few dollars.

Cost: Defensive papers. The most typical way to measure the cost in defensive papers is via the 'tradeoff', i.e., the change in performance *after* applying a given countermeasure (e.g., [151]). We mention two works that make insightful considerations on the 'defensive' cost—or rather, on how the principle of a defense is to increase the 'cost' of the attacker.

- Chen et al. [152] propose to measure the cost of feature manipulation against shallow ML models. The intuition is to carry out the measurements during the *evaluation* of ML models, with the underlying principle of "if an attacker wants to evade an ML model by doing a, they have to spend c_a; and if they want to do b, they have to spend c_b." We remark, however, that [152] does *not* evaluate any attack: the main contribution is measuring the (computational) effort in manipulating some features. (which is why we do *not* include [152] in our Table IV.)
- Heisenhofer et al. [153] acknowledge that attackers can use adversarial examples to evade a ML system for automated speech recognition. Hence, they aim to increase the cost of the attacker to craft such adversarial examples.

An alternative formulation of the "raise the attacker's cost" is intrinsic of papers that focus on the *detection* of security violations. We observe, however, that this detection can occur at different *granularities*. For instance, some papers aim to detect individual adversarial examples (e.g., [93]) or poisoned ML models (e.g., [96]); whereas others have a broader scope and aim to detect attackers (e.g., [102]).

F. Answers to G5 and G6: Data-type and Source-code

We now turn the attention to G5 and G6, which are strongly related to the *reproducibility* [154] of research. We make a statement on this subject at the end of our main paper (§V-D).

G5: Data-type. Past research investigated a plethora of application domains of ML, which we divide in four categories. Three correspond to the broad areas of *images* (e.g., computer vision), *text* (e.g., natural language processing), *audio* (e.g., automated speech recognition); the fourth includes all *other* domains (e.g., malware). These results are in Fig. 10; some papers (e.g., [99]) consider multiple domains, which is why the y-axis goes above 1. More details are shown in Table IV.

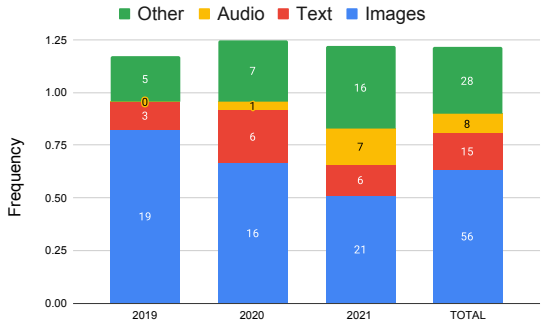


Fig. 10: What are the *data-types* considered in the evaluation?

We can see an interesting trend from Fig. 10: in 2019, over 75% of the papers focused on images (typically using well-known datasets such as MNIST and CIFAR), whereas the percentage dropped to 50% in 2021. The audio domain has also been neglected in 2019, but became prominent in 2021. In particular, papers of 2021 covered much more 'novel' domains, such as graphs [155] or games [156].

G6: Source-code. We looked for links pointing to the source-code of the experiments carried out in all our considered papers. The results are in Fig. 11.

We can see an encouraging trend, as 75% of papers did not release their source-code in 2019, whereas only 35% did not do so in 2021. To facilitate future research, we provide also the actual links to every open code repository of each paper in

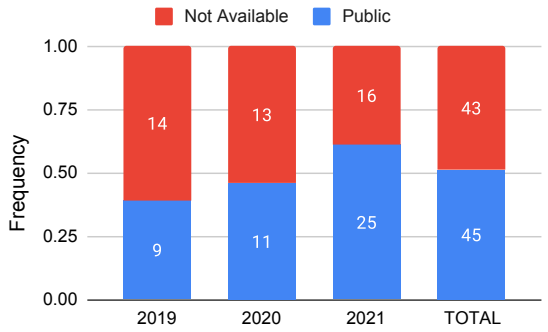


Fig. 11: Has the *source-code* been publicly released?

Table IV. Nonetheless, we stress that even those papers that do not make their evaluation publicly accessible had *valid reasons* for doing so. For instance, [54] describes plenty of details, uses well-known public datasets, and mostly rely on *existing* and publicly accessible code. On the other hand, the experiments in [147] are based on *sensitive data* that cannot be disclosed.

G. Answers to G7 and G8: pipeline and type of ML system

The last generic questions pinpoint the degree of 'realism' considered in the evaluations of prior work. Answering these two questions was not simple: the authors who took part in this activity had several debates before reaching a consensus.

G7: processing pipeline. This question was among the hardest to answer during our survey. Our goal was determining whether a given paper (i) envisioned a ML system, and (ii) developed more components of the ML system than just an ML model. For example, if a paper considered a ML system (by, e.g., providing a schematic), but the evaluation spanned over a single ML model, the answer to G7 was negative. The answer was also negative if the paper made some preliminary checks to the inputs of the ML model (e.g., to ensure that the samples do not violate domain constraints [157]). Simply put, to answer G7 we embraced Abdullah et al. [145] statement, which we quote: "Processing pipelines for modern systems are comprised of signal preprocessing and feature extraction steps, whose output is fed to a machine-learned model. Prior work has focused on the models, using white-box knowledge to tailor model-specific attacks." Answers to G7 are in Fig. 12.

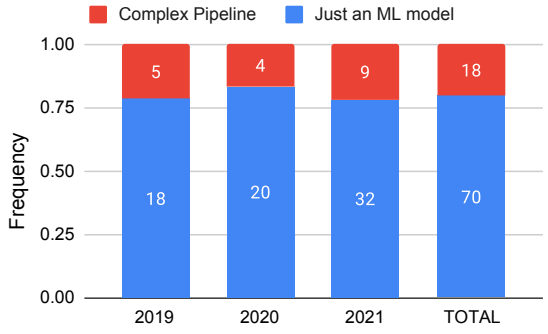


Fig. 12: Has a complex *pipeline* been reproduced in the evaluation?

As expected, nearly 80% of papers perform their attacks by considering a single ML model. Even some papers that attack MLaaS (e.g., [158]) ultimately simply have the MLaaS train a given ML model, and then attempt to violate this model without reproducing a real pipeline. Nonetheless, there are some notable efforts, such as [49] and [22]. These works reproduce a custom ML system by developing *also* a preprocessing component: this component represents an additional layer that an attacker must bypass to reach the actual ML model that will be ultimately targeted. However, we find it concerning that the overall trend shown in Fig. 12 is *stable*. We strongly endorse future work to start developing more components

G8: type of ML system. This question is split in two parts. First, we asked ourselves whether, in any part of the evaluation of a given paper, there was an ML model that can be considered as 'deployed in practice'. For example, papers that attack commercial ML systems (e.g., [25]) automatically fall into this category; however, we also considered papers that used 'extremely popular' ML models (e.g., Yolo [159]) that have been shown to have realistic applications besides benchmarking. The answers to the 'first' part of G8 are shown in Fig. 13. Conversely, if a paper used some 'popular' techniques as a form of preprocessing but the attacked ML model is trained on benchmarks, then the answer to our question was negative—this is the case, e.g., of [49]. Then, if the answer to the first part of G8 was positive (i.e., the paper attacked a 'deployed' ML system), we proceeded to identify the type of (real) ML system. Recall (§II-A) that there exist two main types of ML systems: OPEN and CLOSED (and also INVISIBLE). The answer to the second part of G8 is in Table IV.

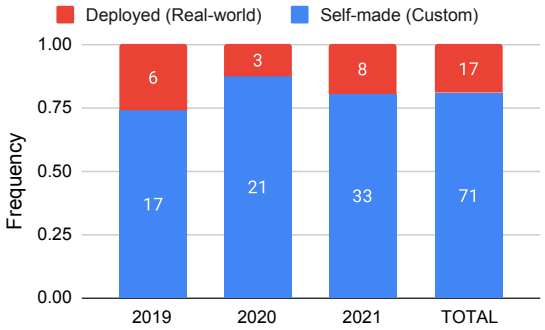


Fig. 13: Does the paper consider an ML model *deployed* in the real world?

By observing Fig. 13, we are not surprised to find out that over 80% of the papers attack ML models (or ML systems) that are entirely self-developed—either by using private data, or by using well-known benchmark datasets. We conjecture that this is due to the difficulty of researchers to acquire permission to use 'commercial' ML systems for research. For example, the notable work by Nassi et al. [25] clearly explains what the authors had to do in order to attack the ML system for object recognition integrated in real autonomous cars. We commend the work by Nassi et al. [25], but similar experiments may be "outside the reach" of most researchers.

Nevertheless, a detailed look at Table IV reveals that, among

those papers that attack real ML systems, there is not a single one which is INVISIBLE. This makes sense: such systems are typically protected by NDA and researchers can hardly use them for experimental purposes. We hope that our position (§V-C) will promote practitioners to collaborate more actively with researchers, so that future works can assess the robustness of real ML systems against against security violations.

H. Answers to the Threat-Model Questions

There is a remarkable lack of clarity in the way that research papers present their threat models (as also hinted in §IV-D).

T1: Parameters. We focused on determining whether the attacker knew anything about the ML model itself, such as the underlying algorithm (e.g., a Decision Tree or a Neural Network), its initial configuration (e.g., the learning rate or the architecture of the neural network) or its learned weights. However – for those papers in which the attacker was assumed to have some knowledge on the ML model – it was difficult to specifically determine what was known. Hence, we considered three possible answers to this question: *full*, *zero* and *intermediate* knowledge. An exemplary case of the latter is the work by Tang et al. [96], stating that their ("black-box") attacker: "does not have information about the inner parameters of the target model [but] he knows the target model's architecture, used optimization algorithm and hyper-parameters.".

T2: Semantics. We consider whether the attacker was aware of the input data-type received by the ML model. For instance: does the attacker *know* that the ML model analyzes images/malware/text? The answer was binary: *yes* or *no*. In most cases, the answer was positive. However, in some rare occurrences the attacker was also oblivious of this information. An example is [110], in which the attacker simply does not need this knowledge because the manipulation affects the ML system at the hardware layer.

T3: Output. The main difficulty we encountered when answering this question was the *format* in which the attacker received the output of the ML system. As highlighted by Jagielski et al. [107], an attacker can receive many forms of feedback by a given ML system. For simplicity, we considered four answers: *decision* (e.g., [12]) if the attacker can only observe the decision of the whole ML system; *label* (e.g., [160]), if the attacker received the class with the highest probability; *probability*, if the attacker received any additional information beyond the label (e.g., [105]); as well as *none*, when the attacker did not receive any feedback (or was not needed).

T4: Training. As stated in our main paper (see §IV-D), identifying if the attacker had some knowledge on the training data was very confusing. To be as specific as possible, we considered the following answers: *full*, if the attacker had full access (read and write) to the training data; *read*, if the attacker can observe the entire training set; *subset* (e.g., [52], if the attacker had a subset of the actual training dataset; *surrogate* (e.g., [146]), if the attacker had samples not included in the training data, but having the same distribution; *distribution* (e.g., [161]), if the attacker only knows the class distribution of the training data; and *none* if the attacker knows nothing.

Remark: Quantitative analyses that take into account the cumulative answers to T1 to T4 are not possible: every paper ultimately considers a different scenario. Therefore, we refrain from deriving any 'trend' from T1 to T4. Considerations should be made on a paper-by-paper basis, and we discuss some of them in the main paper (§IV-B and §IV-C). The individual answers to these questions are provided in Table IV.

I. Complete Table

We summarize our complete literature review in Table IV. This table[11] lists each of the 88 papers considered in our analysis, presented in temporal order: from 2019 until 2021, and from NDSS (typically held in January) until CCS (typically held in November). For each paper, we report the first author name, as well as concise answers to our 12 questions.

Generic Questions (G1 to G8). Most of these are straightforward. For readability, every 'negative' answer is denoted with a blank cell. For G1 (focus), an asterisk denotes a paper whose main contribution is a ML-based solution to a given problem, which is *also* evaluated in adversarial scenarios (e.g., [142]). For G4 (cost), we use: ✗ when no consideration is made; ◐ when it is just mentioned; and ✓ when some measurement is performed. For G5, we also provide the specific data-type (e.g., malware) in the 'other' column (if there are more than one 'other' data-type, we report '+'). For G6 (source code), the ✓ is a *link* that leads to the actual repository. For G7, we use ✓ if the paper develops an actual pipeline, i.e., at least another component besides the ML model (an asterisk denotes attacks at a different layer of the pipeline—e.g., hardware [110]).

Threat-Model Questions (T1 to T4). For T1 (parameters), we use ✗ if nothing is known, ✓ if the ML model is fully known, and ◐ if the attacker knows a portion of its architecture, or has a surrogate model. For T11 (output), we use l if it is the actual label, p if it is a probability, s if it is the decision of the whole ML system, and ✓ if it is not needed. For T11 (training data), we use: ✓ if the attacker has complete read and write access; R if they have complete knowledge of the training data; $\subset$ if the attacker has a subset of the training data; and S if they have a surrogate dataset of the same distribution; and D if they only know the distribution of the training set; ✗ if they know (and can do) nothing about the training data. However, if the answer to T2 is 'poisoning', it is implicitly assumed that the attacker always has some form of write access to the training data (the only exception is [95]).

[11] We note that some (actually, most) papers envisioned attackers conforming to diverse threat models. Due to some of statements made in our main paper (V-B), we show in Table IV the assumptions that correspond to *weakest* attackers (e.g., if a paper considers both a black- and white-box attacker, we will report the description of the black-box attacker in Table IV). The motivation is that attacks stemming from weaker adversaries tend to be more representative of the real world. Moreover, some papers consider dual attacks (e.g., [44] and [162]). In these cases, we only report the one that is given a greater emphasis in the paper.

TABLE IV: The 88 papers considered in our analysis. Each column reports the answer to one of the 12 research questions we used during our survey. If available, the **G6** column provides the *hyperlink* to the websites hosting the source-code of a given paper. Explanations are in Appendix B-I.

Year (subs)	**Venue** (subs)	**Paper** (1st author)	**G1** *Focus*	**G2** *Attack*	**G3** *Paradigm*	**G4** *Cost*	**G5** (Evaluation Data) *Img*	*Text*	*Audio*	*Other*	**G6** *Code*	**G7** *Pipeline*	**G8** *Type*	**T1** *Param.*	**T2** *Sem.*	**T3** *Output*	**T4** *Training*
2019 (23/435)	NDSS (4/89)	Salem [158]	atk	Member.	DL	◐	✓	✓			✓		CLOSED	✗	✓	p	✗
		Li [138]	atk	Evasion	DL	✓	✓				✓			✓	✓	✗	R
		Ma [163]	def	Evasion	DL	◐	✓							✓	✓	p	✗
		Li [161]	atk	Evasion	DL+SL	◐		✓				✓	CLOSED	✗	✓	p	D
	SP (3/84)	Ling [132]	def	Evasion	DL	✓	✓				✓			✓	✓	p	✗
		Wang [164]	def	Poison.	DL	✓	✓				✓			✓	✓	✗	✓
		Nasr [100]	atk	Member.	DL	✗	✓			Finance				✗	✓	p	D
	SEC (6/113)	Tong [114]	def	Evasion	DL+SL	✓				Malware	✓			✗	✓	p	✗
		Demontis [44]	atk	Evasion	DL+SL	✗	✓			Malware				✗	✓	p	S
		Xiao [22]	atk	Evasion	DL	✗	✓					✓	CLOSED	✗	✓	p	✗
		Quiring [165]	atk	Evasion	DL+SL	✗		✓						✗	✓	p	✗
		Hong [110]	atk	Evasion	DL	✗	✓					*		✗	✗	p	✗
		Batina [111]	atk	Stealing	DL	◐	✓					*		✗	✓	p	✗
	CCS (10/149)	Song [166]	atk	Member.	DL	✗	✓				✓			✗	✓	p	✗
		Jia [167]	def	Member.	DL	✗	✓			+				✗	✓	p	✗
		Co [101]	atk	Evasion	DL	✓	✓							✗	✓	p	✗
		Liu [168]	def	Poison.	DL	✓	✓							✓	✓	p	✓
		Baluta [169]	def	Poison.	DL	◐	✓				✓			✓	✓	p	✓
		Zhao [105]	atk	Evasion	DL	✗	✓				✓		OPEN	✗	✓	p	S
		Tramer [12]	atk	Evasion	DL	◐	✓				✓	✓	CLOSED	✗	✓	s	✗
		Wang [170]	atk	Evasion	DL	✓				Graphs				✗	✓	✗	⊂
		Yao [52]	atk	Poison.	DL	◐	✓							◐	✓	p	⊂
		Yang [171]	atk	Stealing	DL	◐	✓						CLOSED	✗	✓	p	D
2020 (24/470)	NDSS (2/88)	Aghakhani [142]	atk	Evasion*	DL+SL	✗				Malware	✓			✗	✓	✗	D
		Yu [34]	atk	Stealing	DL	✓	✓						CLOSED	✗	✓	p	D
	SP (4/104)	Schuster [172]	atk	Poison.	DL	✓		✓						✗	✓	✗	⊂
		Pierazzi [49]	atk	Evasion	SL	✓				Malware	✓	✓		✓	✓	p	R
		Chen [88]	def	Evasion	DL	✓	✓				✓			✗	✓	l	✗
		Jan [147]	atk	Evasion*	DL	✓				Network				✗	✓	p	⊂
	SEC (8/157)	Salem [146]	atk	Member.	DL	✗	✓			Location				◐	✓	p	S
		Chandrasekaran [148]	atk	Stealing	DL+SL	✓	✓	✓		+				◐	✓	p	✗
		Suya [103]	atk	Evasion	DL	✓	✓				✓			✗	✓	p	S
		Jagielski [107]	atk	Stealing	DL	◐	✓							✗	✓	l	D
		Quiring [173]	atk	Evasion	DL	✗	✓				✓	✓		✗	✓	p	✗
		Li [151]	def	Evasion	DL	✓		✓						✗	✓	p	D
		Leino [174]	atk	Member.	DL	✓	✓			+				✓	✓	p	✗
		Zhang [175]	atk	Evasion	DL	✗	✓							✓	✓	p	✗
	CCS (10/121)	Nassi [25]	atk	Evasion	DL	✗	✓				✓	✓	CLOSED	✗	✓	s	✗
		Li [176]	atk	Evasion	DL	✗			✓					✓	✓	p	⊂
		Shan [102]	def	Evasion	DL	◐	✓				✓			✓	✓	p	✗
		Pang [162]	atk	Poison.	DL	✓	✓				✓			✓	✓	p	✓
		Abdelnabi [177]	atk	Evasion*	DL	✗	✓			Phishing	✓	✓		✗	✓	✗	✗
		Li [178]	atk	Evasion	DL	✓	✓							✓	✓	p	✗
		Lin [179]	atk	Poison.	DL	✓	✓	✓			✓			✓	✓	✗	✓
		Chen [180]	atk	Member.	DL	✓	✓							✗	✓	p	D
		Zanella [24]	atk	Member.	DL	✓		✓			✓			✗	✓	p	✗
		Song [23]	atk	Member.	DL	✓		✓						✗	✓	p	S
2021 (41/644)	NDSS (3/87)	Hui [104]	atk	Member.	DL	✗	✓			+	✓			✗	✓	p	✗
		Huang [181]	atk	Poison.	DL	✓				Ratings				✗	✓	✗	⊂
		Barradas [90]	atk	Evasion*	SL	✓				Network	✓			✗	✗	✗	✗
	SP (5/115)	Xu [149]	def	Poison.	DL	✓	✓		✓		✓			✓	✓	p	✓
		Abdelnabi [182]	atk	Evasion	DL	◐		✓			✓	✓		✗	✓	p	✗
		Chen [91]	atk	Evasion	DL	✓			✓		✓	✓	(both)	✗	✓	s	S
		Abdullah [145]	atk	Evasion	DL	✓			✓			✓	CLOSED	✗	✓	✗	✗
		Nasr [35]	def	Evasion	DL	✓	✓			Finance				✗	✓	p	✗
	SEC (24/246)	Sato [28]	atk	Evasion	DL	✓	✓				✓	✓	OPEN	✓	✓	p	✗
		Nasr [89]	atk	Evasion	DL	✓				Network	✓	✓		✗	✓	p	D
		He [183]	atk	Member.	DL	◐				Graph				✗	✓	p	✗
		Severi [184]	atk	Poison.	DL+SL	✓				Malware	✓			✗	✓	p	✓
		Bagdasaryan [95]	atk	Poison.	DL	✓	✓	✓			✓			✗	✗	✗	✗
		Xi [155]	atk	Poison.	DL	✓				Graph	✓			◐	✓	✗	S
		Tang [96]	def	Poison.	DL	✗	✓				✓			◐	✓	p	⊂
		Schuster [185]	atk	Poison.	DL	✓		✓					OPEN	✗	✓	✗	⊂
		Carlini [54]	atk	Poison.	DL	✓	✓							✗	✓	✗	⊂
		Vicarte [186]	atk	Poison.	DL	✓	✓							✗	✓	p	✗
		Lovisotto [150]	atk	Evasion	DL	✓	✓				✓		OPEN	✗	✓	✗	⊂
		Carlini [30]	atk	Member.	DL	✓		✓				✓	OPEN	✗	✓	p	✗
		Han [97]	atk	Evasion*	DL	✗				Graph				✗	✓	p	✗
		Eisenhofer [153]	def	Evasion	DL	✓			✓		✓	✓	OPEN	✓	✓	p	R
		Wu [156]	atk	Poison.	DL	✓				Games	✓			✗	✓	s	✗
		He [187]	atk	Stealing	DL	✓	✓							✗	✓	l	S
		Rakin [112]	atk	Evasion	DL	✓	✓				✓	*		✗	✓	p	✗
		Jia [188]	def	Stealing	DL	✓	✓		✓		✓			◐	✓	l	⊂
		Zhu [189]	def	Stealing	DL	✗	✓					*		✗	✓	p	✗
		Xiang [190]	def	Evasion	DL	✓	✓				✓			✓	✓	p	✗
		Lin [191]	atk	Evasion *	DL	✓	✓			Phishing	✓	✓		✗	✓	p	✗
		Azizi [192]	def	Poison.	DL	✗		✓			✓			✓	✓	p	✓
		Hussain [93]	def	Evasion	DL	✗			✓		✓			✓	✓	p	✗
		Song [99]	def	Member.	DL	✓	✓			+	✓			✗	✓	l	✗
	CCS (9/196)	Zheng [92]	atk	Evasion	DL	◐			✓			✓	CLOSED	✗	✓	✗	✗
		Mu [193]	atk	Evasion	DL	✓				Graphs				✗	✓	l	✗
		Bahramali [194]	atk	Evasion	DL	✓				Network				✗	✓	✗	S
		Sheatsley [157]	atk	Evasion	DL	✗				Network				✓	✓	p	R
		Du [94]	def	Evasion	DL	✓	✓	✓						✓	✓	p	R
		Li [195]	def	Evasion	DL	✓	✓				✓			✗	✓	✗	✗
		He [196]	def	Member.	DL	✗	✓				✓			◐	✓	l	S
		Li [160]	atk	Member.	DL	✓	✓				✓			✗	✓	l	✗
		Chen [197]	def	Member.	DL+SL	✗	✓			+	✓			✗	✓	p	✗
Year (subs)	**Venue** (subs)	**Paper** (1st author)	*Focus* **G1**	*Attack* **G2**	*Paradigm* **G3**	*Cost* **G4**	*Img* **G5** (Evaluation Data)	*Text*	*Audio*	*Other*	*Code* **G6**	*Pipeline* **G7**	*Type* **G8**	*Param.* **T1**	*Sem.* **T2**	*Output* **T3**	*Training* **T4**