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

***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 these two domains. To this end, 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 and highlight positive trends and blind spots. Finally, we state positions on precise and cost-driven threat modeling, collaboration between industry and academia, and reproducible research. If widely adopted, these positions 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**

I. Introduction

Several recent surveys indicate that protecting 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 retraining* [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] showcase successful security violations of ML models by means of sophisticated strategies such as gradient-based algorithms.

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 security practice is real and identify several underlying causes. We aim to reduce this gap by proposing guidelines for future research that reflect our observations of aspects overlooked by, or inconsistent within, much of the literature.

To reach our goal, we first revisit fundamental aspects 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 binding *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 of real-world scenarios (§III) that elucidate: (a) the *architecture* of a complex ML system deployed in a large online social network, (b) the role played by *human effort* in devising sophisticated attacks to win an acclaimed ML evasion competition, and (c) the *lack of evidence* of adversarial examples in the context of a commercial-grade ML system for phishing webpage detection.

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 either attacks against ML or corresponding defenses. Of these papers, 89% only consider algorithms based on neural networks, 63% focus on computer vision, and 80% perform their evaluation on "benchmark" data sets. We also discover several inconsistencies in the definitions of the threat models presented in reputable prior work. On the other hand, we identify several *positive trends*, such as an increasing amount of research that considers "weaker" 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 taking four positions that both researchers and practitioners in ML security can adopt to close the gap between these two domains (§V). In particular, we encourage the community to (a) adapt threat models from ML 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 the detailed findings of our survey of recent security papers—found in the Appendix.

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" [16]. At its core, the process entails applying a *learning algorithm* to *training data*, which yields a *machine learning model*. The purpose of the ML model is to generalize to new data: given an *input*, the 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 [16], [17]. The effectiveness of the ML model is measured during a *validation phase*, during which the ML model analyzes test data and its predictions are compared to some known *ground truth* (e.g., labels [18]). 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 [17] are denoted as *deep learning*, whereas others are broadly denoted as *shallow learning* and include, e.g., SVMs and tree-based algorithms [19].

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.

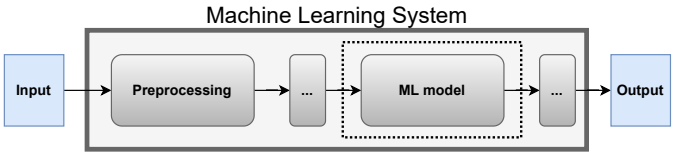


Fig. 1: Example ML system. The system receives an input, which is preprocessed and then fed to an ML model, the results of which may be further processed before providing the system's final output.

[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 [20] or word embeddings [21]). Some ML systems require constant updates, which can be done in an automated fashion [22]. Others may adopt company-specific guidelines [23]. A given input may not even reach any ML model integrated in the ML system. This is typical, e.g., in Network Intrusion Detection Systems (NIDS), wherein samples matching known "signatures" are put in a different pipeline that does not include any ML model [24].

> 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 that distinguish the relationships (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 cannot inspect (i) the underlying source code, and (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 [25] GPT2 [27]	Autonomous Driving Natural Language	Sato [26] Carlini [28]
Closed	Google Translate [29] ClarifAI [31]	Translation (*unrestricted*) MLaaS (*restricted*)	Pajola [30] Yu [32]

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 have visibility of the output (e.g., via an API [33]). 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, but it may be a result of other systems or it may arrive after a very long and 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, and we will present a case study in §III-A.

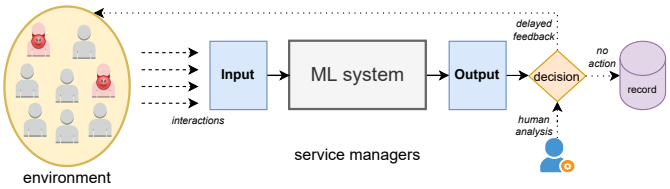


Fig. 2: An INVISIBLE ML system. The low privileged users in the *environment* interact with the ML system, the output of which can be used in diverse ways by its *managers*. For example, the alarms of a NIDS may be inspected by sysadmins and then used to block suspicious hosts; whereas a network management 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 investigated the security of machine learning, a research field typically denoted as "adversarial machine learning" [34]. Although work in this field has existed since the early 2000s [35], adversarial ML became popular when Szegedy et al. [36] discovered that DL was vulnerable to *adversarial examples*. In simple terms, adversarial examples are "similar" to "normal" inputs but induce a given ML model to behave incorrectly [37]. 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 [38]. (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. [34] was among the first to formalize the potential threats to ML. Since then, more efforts followed—some more general (e.g., [37], [39]), others more specific (e.g., [40], [41]). 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 [42]. They can also, however, attempt to "steal" the ML model [43], or extract private data from it via *membership inference* [44] attacks that infer whether a person's data is in the model's training set.

KNOWLEDGE. Srndiç and Laskov [41] 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 [39]: "white-box," when the attacker knows *everything*; "black-box," when the attacker knows *nothing*; and "gray-box," for intermediate cases [40].[2]

[2]Such color-based terminology may be offensive to 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 such terminology (§V).

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 [45]; they can control inputs to a (trained) ML model and observe its output [46]; and/or they can operate from diverse spaces of the data pipeline of a ML system (the so-called "problem/input" and "feature" spaces [47]). In some circumstances [48], 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 [42].

STRATEGY. To reach their GOAL, attackers exploit their KNOWLEDGE and CAPABILITIES and enact a STRATEGY, e.g.: an attacker that fully knows a deep learning model can use its gradient to craft adversarial examples [49]; attackers who can manipulate training data may introduce a *backdoor* [50].

We observe, however, that the terminology above has a limitation: it focuses exclusively on the ML **model** (the inner and 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 works 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." [51]

> OBSERVATION: Despite abundant evidence 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" [52] (the actual quote is from a renowned con artist [53]). Given a sufficient amount

of resources, any attacker can succeed in their goal [54]. Indeed, the purpose of security mechanisms is to **raise the cost sustained by the attacker** [55], 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., include implementation efforts, preliminary evaluations, maintenance, and computational resources [10], [56]) that must be factored in to any decision about whether to build or deploy them.

Cybersecurity in practice. Let us substantiate our previous claims with public statements made by cybersecurity professionals. On July 22nd, 2022, a tweet [57] by Sophos AI's 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 [58] 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*: [59])

Berlin's further statements mentioning that "Sophos would block most adversarial ML attacks within a few hours, making them useless", and conclude with: "Given the existence of an already large number of more prevalent attacks that do bypass detections, why prioritize this one?"

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

D. Related Work

Many papers share our goal of improving the security of real-world ML systems. The authors of [14] and [60] 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. On the other hand, most literature surveys (e.g., [37], [39], [61]) focus just on the research perspective. We consider both perspectives, because we aim to bridge the existing gap between these two sides.

We briefly summarize two closely related research areas that are orthogonal to this paper. **(1)** The performance of ML tends to degrade over time [62]-[63]. 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. **(2)** ML can also be used as an attack vector (e.g., "deepfakes" [64], or "attribute inference" [65]). This research area (i.e., "offensive ML") is complementary to ours, because we focus on the protection of ML systems.

To the best of our knowledge, this is the first position paper (on ML security) whose main positions 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)

In what follows, we present several case studies contributed by industry practitioners. Our intent is to shed light on practical issues that are obscure to many researchers—but which epitomize operational cybersecurity.

The first (§III-A) presents the architecture of a real INVISIBLE ML system deployed in an online social network (OSN). The second (§III-B) sheds light on the human effort required to devise query-efficient attacks during an ML evasion challenge organized by renowned tech companies. The third (§III-C) describes the triaging of phishing webpages that were poorly classified by the ML detector of a security company. We also provide a fourth one (Appendix A-B) illustrating the simplicity of evading malware detectors.

Because all of our case studies entail ML systems for *detection* tasks, this section will primarily focus on attacks commonly known in the literature as "evasion attacks," i.e., misclassifications at test time.

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

This case study reflects the experience of 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 a social network; for example, they want to post a pornographic image with some text, which may lure a user to click on an embedded URL. The OSN 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. [66], [67]). 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., [68]). Such a "black-box" scenario (which has also been envisioned in its "white-box" variant [69]) 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, as explained in §II-A. In fact, content classification (block/allow) is the last line in a series of defenses erected by this OSN. A typical OSN uses an entire funnel of countermeasures that work in concert to provide defense at different layers: to be successful, an attacker must evade all of them. 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).

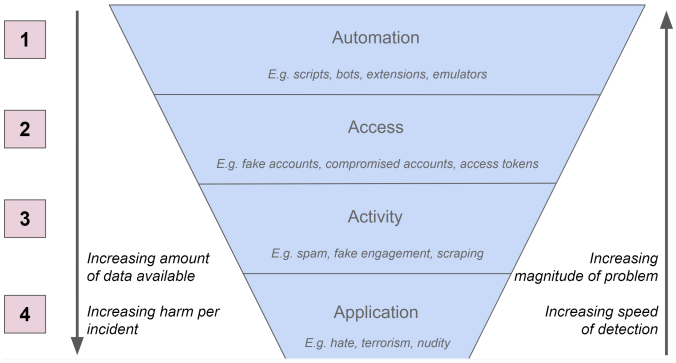


Fig. 3: Example of a real ML system for spam detection in OSN. The ML system consists in a 'funnel' of four interconnected defensive layers, each with its own logic. The attacker must bypass all such layers to be successful.

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, OSNs are constantly targeted by bots and automated scripts [70]. The AUTOMATION layer automatically prevents "black-box" evasion strategies based on repeatedly querying the system over short time frames.

2) ACCESS *layer.* Automation detection is the only viable defense of the OSN against logged-out reads at scale. In contrast, an account is needed for writes (e.g., *posting* content on the OSN) or logged-in scraping. Most malicious activity is perpetrated by fake or compromised user accounts [71]. 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 made to the OSN and determine whether the corresponding write permission was obtained legitimately by the requesting entity.

3) ACTIVITY *layer.* If the OSN cannot stop malicious actions just for being automated (in the AUTOMATION layer), and cannot catch bad actors when they obtain access (in the ACCESS layer), then the next opportunity is when they begin carrying out their (malicious) activity on the OSN. To reach their goal, spammers and other abusers must act in fundamentally different ways from normal users. Such differences include how they interact with the social graph [72], the distribution of the actions undertaken on the OSN, 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 the OSN's policies, can be unintentionally produced by "benign" users? To address these concerns, the OSN has a fourth, APPLICATION layer. Assuming that the above layers are mostly effective (i.e., low false positives/negatives), most of the violations in this layer will be small-scale and driven by real users. This is the layer where ML can really shine, such as nudity detectors and hate-speech classifiers—i.e., the DL models typically targeted in research papers [69].

Complexity. A real 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 such components to be successful.

B. In-Depth Analysis of ML Evasion Competitions

Since 2019, the Machine Learning Security Evasion Competition (https://mlsec.io) has incentivized contestants to evade ML-based detectors, attracting enthusiasts and practitioners [73]. In what follows, we analyze the 2021 Anti-Phishing Evasion Challenge, whose task was to create a set of phishing webpages that evade a suite of ML detectors.

Rules. Contestants were given 10 phishing webpages (as HTML) and were asked to manipulate them in such a way as to preserve the original rendering while evading seven ML-based detectors. These detectors are CLOSED ML systems: contestants could input their manipulated webpages to the detectors, which provided a 'phishing' probability (within [0–1]) as output. Winners were determined on two criteria: the number of successful evasions, and the number of queries made to the detectors. Contestants could submit an unlimited number of *solutions*, i.e., sets of manipulated webpages to be reviewed by the organizers at the end of the challenge. A public leaderboard 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 and lasted until Sept. 17 (42 days in total).

Results (high-level). Four teams were able to craft variants of all 10 phishing webpages that evaded all 7 detectors, thereby achieving a perfect score of 70 points. These four teams, in order of ranking, made 320, 343, 608, and 9982 queries [74]. The 1st-place team (two scientists from Kaspersky) and 3rd-place team (a single individual, Ryan Reeves) teams both published their methodologies(320 queries) [75], [76] 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 have often done, e.g., [32]), leads to the conclusion that the winning solution had the lowest cost: only 320 queries, i.e., an improvement of 47% over the third-place solution (608 queries) and of 97% over the 4th place team (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 displayed in Fig. 4. Here, a "submission" corresponds to an upload of a set of manipulated phishing webpages for review[3]. In other words, we can assume that the last submission made by each team corresponds to the point in time in which they "finished" their attack.

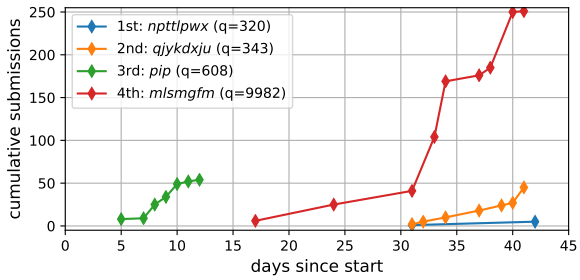


Fig. 4: Temporal distribution of cumulative submissions during the phishing MLSEC in 2021 (started on Aug. 6th). Each line indicates a team (q=queries).

The 3rd-place team (green line in Fig. 4) was the first to arrive at a perfect evasion, requiring 608 queries distributed over eight days, 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. 4) 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 adopted a "wait-and-see" approach [75] that allowed them to take into account the results of other participants. Specifically, their initial intention was to replicate typical "black-box" techniques (e.g., [77]) based on model replication, which are query-intensive. However, they changed their mind 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. 4) 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 effort.

Put simply: the 1st-place team won the challenge by taking advantage of the scoring method. Their strategy required fewer queries, but also more time (in absolute terms) and, likely, more human effort.

> OBSERVATION: Measuring attack efficiency with query counts alone does not fully capture attacker effort. **Human effort** in attacks can be significant.

As an additional contribution, we publicly release the code [78] and data [79] of our analysis, for which we received explicit permission by the organizers of MLSEC.

C. 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 [80], which contains over 1,600 reports of AI failures (as of August 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 that are 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*. In contrast to the previous case study, however, here 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[4]) consists of an ensemble of image-based 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 [81]: For each input, the output is a 'phishing confidence' score. Besides using such score to make a decision (i.e., block/allow), security analysts use such score to improve the ML system, by triaging 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. To identify adversarial examples on the web, we searched for them among the 40 domains that (according to the cybersecurity company) were most commonly targeted[5] by phishing campaigns in July 2022. During this timeframe, the ML system analyzed hundreds of thousands of inputs—among these, 9,174 were flagged as 'uncertain' by the ML system, and required manual triage. We used these as a starting point, and we began to study all such samples (i.e., screenshots). To make our in-depth analysis humanly feasible, we visually inspected each sample asking ourselves "can this be an adversarial example?", and decided to stop once we

[3] An even more fine-grained analysis would study the temporal distribution of individual API calls. Unfortunately, such data was not logged.

[4] 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.

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

obtained a positive answer for 100 samples. We reached this number after going through 4,600 samples (half of our initial population): the discarded ones had nothing in common with adversarial examples, due to being "out of distribution" [82]—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 reasons that induced the ML system to be 'uncertain' and, in particular, if such reason could be traced back to gradient-based techniques typical of adversarial examples. To reduce bias, this entire process (first inspection and in-depth analysis) entailed 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 strategies—which are often effective. We show examples of these attacks in Fig. 5 and quantify their prevalence in Table II. The most prevalent attacks include cropping, masking, stretching, 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 [83] and can (still) ably fool ML systems without relying on gradient computations. We found that these methods typically target the *company name* when it is part of a logo, mostly by attempting to manipulate the logo's visual representation. We also found samples employing a different logo altogether, as well as and "company name style" attacks that alter the company name portion of the logo. 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 that alter the background, especially behind company logos, to confuse the ML system.

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		

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 [84], [85]). 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. 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.

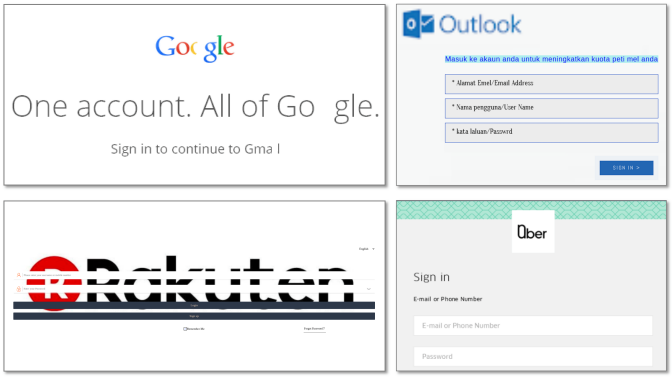


Fig. 5: Four evasive phishing samples, depicting the use of masking, cropping, blurring, and misspellings to disrupt the detection of company names, logos, and login-related keywords (e.g., "Passwrd").

> OBSERVATION: attackers attempting to evade phishing ML detectors employ tactics relying on simple, cheap and effective methods—which can be hardly linked with gradient computations.

As an additional contribution, we publicly release the 100 evasive webpages that we considered for our analysis [86].

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 negative trends in inconsistent terminology adopted in research (§IV-D). A detailed description of our study is provided in Appendix B.

> **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. Inspired by Arp et al. [14] and Apruzzese et al. [81], 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 identify 88 papers within our scope (shown in Table IV), which we then analyze according to the methodology detailed in Appendix B.

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% consider "evasion" attacks (i.e., cause misclassification at "test-time");
- 89% rely only on deep learning algorithms;
- 56% attempt to measure the cost in some way (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.

We refer the interested reader to the Appendix for additional descriptions, interpretations, takeaways, and figures showing ongoing trends (from 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. [87] 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. [88] 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 [52]). 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 be as unrealistic as attacks stemming from powerful adversaries using simple 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 [46]). We highlight two such papers, both of which consider attacks against ML systems for automated speech recognition (ASR).

- Chen et al. [89] 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 [89]: "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. [90] consider (also) 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 [90]: "[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 vastly rely on their domain expertise (§III-C).

Broader Goals. Some papers envision threat models where the *attack succeeds* despite the ML model's *correct response*. We share two that consider production-grade ML systems:

- Xiao et al. [20] "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. [23] 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 (which are crucial for decision-making in autonomous driving), meaning that these systems can be tricked via "phantom" figures with no spatial depth. The au-

thors 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 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) propose original *defenses*.

Defending against knowledgeable attackers. A common approach in defensive papers is to consider adversaries with perfect knowledge (e.g., [91], [92]). 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. [26]). 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 [93] show how to "poison" an ML system by manipulating its source code, i.e., without any change to the training data.

Defending against attackers with limited knowledge. Tang et al. [94] 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 [95]. Some (e.g., [13], [14]) may argue that similar defenses violate the "Kerchoff principle" [96], but we promote the approach taken in these works ([94], [95]) because attackers without perfect knowledge are a very likely threat in the wild, and hence corresponding defenses demand attention in research. Nevertheless, some works have shown that attackers with limited knowledge may be as dangerous as omniscient ones. In particular, Song and Mittal [97] show that "the gap between white-box attack accuracy and black-box attack accuracy is much smaller than previous estimates [98]."

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

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 terms "white-box" and "black-box," which are very common in the literature, 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 example papers [20], [99]–[101], all of which focus on "evasion" attacks (i.e., missclassifications at "test-time" [37]).

- According to Co et al. [99]: "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 [41].
- Shan et al. [100], 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. [20] 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. [20] 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. [101] assume a "black-box" attacker that "does not have direct access to the target model or knowledge of its parameters", but which "has access to pretrained 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. [99].

We also report the definition of the "gray-box" setting of Hui et al. [102]: "The graybox setting 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. [100]—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. [103] 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. [42]: here the attackers' goal is first presented in terms of "*system* operation", but 10 lines later the knowledge is presented with a "white-box" terminology defined as follows: "the attacker has full knowledge of the target *classifier*". Ad-

ditional confusion stems from the so-called "no-box" attacks envisioned by Abdullah et al. [104], whose definition conflicts with the original one by Li et al. [48]. Specifically, the "no-box" attacker in Abdullah et al. [104] knows *nothing*, which is the de facto assumption of "black-box" attackers. In contrast, Li et al. [48] 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 know 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. [100] 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. [101] 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 line). 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 [105], 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

So far 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	Queries are not always an effective measure of attack cost.	§III-B
6	Evidence of adversarial examples in the wild is scarce.	§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-B

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). Let us discuss each of these positions.

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 the single ML model (§IV-A), 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-B).

> POSITION: Threat models should **precisely** define the viewpoint of the attacker with respect to **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. Let us provide some recommendations.

GOAL+. Real attackers have broader GOALS than just attacking a single ML model. For example, they may want to caus damage to all tenants of an ML system [10], or simultaneously fool humans and ML classifiers [106], [107].

KNOWLEDGE+. To better portray real threats, KNOWLEDGE assumptions should cover all components of ML systems, e.g., any preprocessing steps [47]. For instance, an attacker who has *perfect knowledge* of an ML model, but *zero knowledge* on the preprocessing will have *limited knowledge* of the ML system.

CAPABILITIES+. As with KNOWLEDGE, attacker CAPABILITIES must consider all components of the ML system. For example, attackers should only be assumed 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 [40]). Moreover, the *output* of the ML system must be explicitly defined (e.g., label-only, label-and-score, top-k-scores, all-scores, logits [105]). 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. [108] apply manipulations at the hardware layer, Bagdasaryan et al. [93] manipulate the source code used to develop the ML model, and Batina et al. [109] derive additional information by spying on electromagnetic side-channels. We note that, Rakin et al. [110] describe their threat model (in terms of KNOWLEDGE and CAPABILITIES) both from the system perspective and the ML perspective, which is 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 either from a researcher-to-researcher, or from a researcher-to-practitioner viewpoint. These terms are: BOX, ACCESS, EVASION, ADVERSARIAL. Let us describe and take our stance on each of these terms.

BOX. We have already highlighted the 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/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 be precise (e.g., *read*, or *write*) when categorizing the access—avoiding ambiguous terms (e.g., "direct" or "internal"); and specify the type of access for *every* component of the ML system.[6]

EVASION. In related literature, this term denotes attacks which (i) cause "missclassifications" and which (ii) occur "at test time". In contrast, the generic definition of "evasion" in cybersecurity is to "bypass" a detection mechanism [111], i.e., to cause a malicious sample to be classified as benign. For instance backdoor attacks [50] are considered to be "poisoning" *in literature* [37] and not "evasion". However, from a security standpoint, a (real) attacker does not want to "poison" a dataset—rather, they use "poisoning" (via, e.g., backdoors) to achieve their goal: evade 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 [50], exploiting adversarial examples [112], or also by ignoring the ML model completely [108]).

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 on this term is to use it only when referring to established, well-defined notions,[7] such as *adversarial example* or *adversarial ML*. Nonetheless, let us make an important observation, inspired by [60]: adversarial examples are not malicious per-se; what is malicious is crafting and using an adversarial example. Studying adversarial examples can have applications that are orthogonal to security, such as improving robustness of ML methods [113].

B. Cost-driven Security Assessments

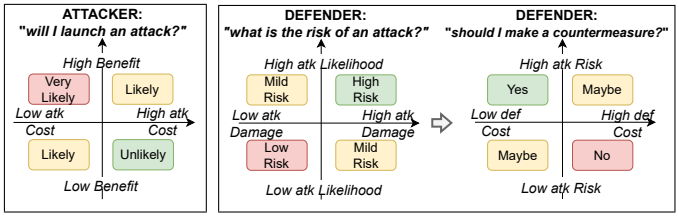
Our second statement is motivated by the poor consideration given by recent papers (§IV-A, but also see Appendix B-E) to the economics of cybersecurity: 28% do not mention cost at all; and although 50% perform some measurements, the human factor—which can be significant (§III-B)—is often neglected.

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

Context. Biggio and Roli [37] 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-C). Real attackers, as pointed out by Wilson et al. [114], 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 [52].

> **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 [115].

Proposal. To promote future research that has a greater impact *in practice*, we extend the recommendations of Biggio and Roli [37] 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 [115]. The designer can then simulate the attack and gauge the corresponding damage, yielding the *risk* [116], [117] of the attack (Fig. 6b, left). Finally, the designer can conceive any given countermeasure, estimate its costs (e.g., implementation, overhead, upkeep), and—depending on the risk of the corresponding attack—assess whether such a countermeasure should be deployed or not (Fig. 6b, right). Simply put, quoting from [60], 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. The most *likely* threats are those that yield high *benefit* but with a low *cost*.

(b) Defender perspective. The defender assesses an attack's *likelihood* and its *risk* by gauging the *damage* (left). On the basis of such risk, the defender will develop a *countermeasure* if its *cost* is acceptable.

Fig. 6: Cost-driven threat modeling. The proactive security lifecycle [37] should include economic considerations for both the attacker and the defender.

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

- Although some papers do include measurements of the "cost" of an attack (or of cost tradeoffs for defense), all such measurements overlook human cost factors (see §III-B). Hence, we endorse future studies *to account for human cost*, such as by reporting the time needed to write the source-code for a given attack/defense (as facilitated by a GIT repository's commit history, for example).

[6] 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.

[7] "Adversarial attack" and "adversarial ML threat model" are both tautologies and can be replaced with "attack against ML" and "threat model."

- It is wrong to consider the attacker/defender battle as a zero-sum game in economic 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) example in Appendix A-C that makes use of the fact that cybersecurity is typically outsourced [118].
- Researchers proposing "novel" attacks should assume the viewpoint of a real adversary. An attack will not be launched if—despite initial success—such an attack can be defused via simple heuristics (refer to §III-A).

We also note that clearly defining the knowledge and capabilities of an attacker with respect to each component of a ML system (§V-A) is greatly beneficial for cost-driven assessments *in research*. In particular, 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.

C. Collaborations between Industry and Academia

Our third position stems from the facts that (i) merely 20% of our analyzed papers (§IV-A) consider real ML systems; and (ii) 90% of papers only consider DL 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 **collaborate more actively** 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. 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. [30], in which specific character modifications induce Google Translate to output a wrong translation. From the viewpoint of the researcher, it is impossible to determine precisely *why* the attack was successful. (E.g., was it due to a problem of the preprocessing phase? Or maybe of the ML model itself?)

We acknowledge that asking technology companies to publicly release their own ML systems is unrealistic (although some examples do exist [119]). 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., [120]), and/or dedicating resources to processing researcher requests (e.g., [121]); in both cases companies benefit by gaining thorough, independent security assessments of their ML systems. Another valuable contribution would be providing high-level schematics of ML systems (as showed in §III-A), which could inspire researchers to run experiments using pipelines that better resemble real ML systems.

D. Just Culture and Reproducible Research

Our last position is motivated by the fact that only 51% of our analyzed papers publicly release their source code (§IV-A). We acknowledge that there may be legitimate reasons that prevent complete disclosure (which we discuss in Appendix A-D). We conjecture, however, that this low number is in large part due to fear of being "blamed."

> POSITION: In ML security, source-code disclosure must be promoted with a **just culture** [122].

A "just culture" [122] 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* [123], which seeks to avoid mistakes by actively blaming the corresponding authors—with the intention of discouraging individuals from making mistakes out of fear of shaming. 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 [124], whose evaluation was found to be flawed *after* its publication [125]. Such flaws, however, were only spotted *because* the implementation of [124] was publicly accessible. Fear of blaming could have induced its authors [124] 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 [124] 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., [126]) to help future research to avoiding mistakes.

VI. CONCLUSIONS

In conclusion, we hope to see (i) more reproducible research, which (ii) adopts threat models on entire ML systems descibed via precise terminology, making (iii) cost-driven assessments factoring the human effort, and that (iv) fosters active collaboration with industry. As a final remark, we also endorse developments of novel techniques for the forensics of ML incidents: perhaps real attackers *do* compute gradients—but we cannot prove it (yet).

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 *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1955–1972.

[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 *Eighteenth Symposium on Usable Privacy and Security (SOUPS 2022)*, 2022, pp. 97–116.

[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: 2022-08-24.

[7] L. N. Tidjon and F. Khomh, "Threat assessment in machine learning based systems," *arXiv preprint 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 *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 17, 2021, pp. 15 458–15 463.

[10] G. Apruzzese, R. Vladimirov, A. Tastemirova, and P. Laskov, "Wild networks: Exposure of 5g network infrastructures to adversarial examples," *IEEE Transactions 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*, vol. 63, no. 9, pp. 1278–1308, 1975.

[12] F. Tramèr, P. Dupré, G. Rusak, G. Pellegrino, and D. Boneh, "Adversarial: Perceptual ad blocking meets adversarial machine learning," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 2005–2021.

[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 Secur. Symp.*, 2022.

[15] "A Complete List of All (arXiv) Adversarial Example Papers." https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html, Aug 2022.

[16] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[18] 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.

[19] 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.

[20] Q. Xiao, Y. Chen, C. Shen, Y. Chen, and K. Li, "Seeing is not believing: Camouflage attacks on image scaling algorithms," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 443–460.

[21] C. Song and A. Raghunathan, "Information leakage in embedding models," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 377–390.

[22] 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 *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 363–375.

[23] 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 *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, 2020, pp. 293–308.

[24] J. Gardiner and S. Nagaraja, "On the security of machine learning in malware C&C detection: A survey," *ACM Comput. Surv.*, vol. 49, no. 3, p. 59, 2016.

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

[26] 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 *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 3309–3326.

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

[28] 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 *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2633–2650.

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

[30] L. Pajola and M. Conti, "Fall of giants: How popular text-based mlaas fall against a simple evasion attack," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2021, pp. 198–211.

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

[32] 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 *NDSS*, 2020.

[33] M. Nasr, S. Songi, A. Thakurta, N. Papemoti, and N. Carlin, "Adversary instantiation: Lower bounds for differentially private machine learning," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 866–882.

[34] 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.

[35] N. Dalvi, P. Domingos, S. Sanghai, D. Verma *et al.*, "Adversarial classification," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, Aug. 2004, pp. 99–108.

[36] 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.

[37] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Elsevier Pattern Recogn.*, vol. 84, pp. 317–331, 2018.

[38] 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.

[39] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Sok: Security and privacy in machine learning," in *Proc. IEEE Europ. Symp. Secur. Privacy*, Apr. 2018, pp. 399–414.

[40] 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.

[41] N. Srndic and P. Laskov, "Practical evasion of a learning-based classifier: A case study," in *Proc. IEEE Symp. Secur. Privacy*, 2014, pp. 197–211.

[42] 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 *Proc. USENIX Secur. Symp.*, 2019, pp. 321–338.

[43] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction {APIs}," in *25th USENIX security symposium (USENIX Security 16)*, 2016, pp. 601–618.

[44] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.

[45] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proc. Int. Conf. Machin. Learning*. Omnipress, 2012, pp. 1467–1474.

[46] 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.

[47] F. Pierazzi, F. Pendlebury, J. Cortellazzi, and L. Cavallaro, "Intriguing properties of adversarial ml attacks in the problem space," in *IEEE Symp. Secur. Privacy*, 2020.

[48] Q. Li, Y. Guo, and H. Chen, "Practical no-box adversarial attacks against dnns," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 849–12 860, 2020.

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

[50] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2041–2055.

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

[52] N. Carlini, "Poisoning the unlabeled dataset of semi-supervised learning," in *USENIX Secur. Symp.*, 2021.

[53] "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.

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

[55] 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.

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

[57] "Why adversarial examples aren't a primary concern at Sophos AI," https://twitter.com/joshua_saxe/status/1550545466072264704/photo/1, Aug 2022.

[58] "Sophos AI: Konstantin Berlin," https://ai.sophos.com/team/konstantin-berlin/, Aug 2022.

[59] "Konstantin Berlin Twitter," https://twitter.com/kberlin/status/1550959458171453440?ref_src=twsrc%5Etfw, Aug 2022.

[60] 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," *Digital Threats: Research and Practice*, 2022.

[61] 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.

[62] 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 *Proc. USENIX Secur. Symp.*, 2017.

[63] 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 *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2327–2344.

[64] D. Güera and E. J. Delp, "Deepfake video detection using recurrent neural networks," in *2018 15th IEEE international conference on advanced video and signal based surveillance (AVSS)*. IEEE, 2018, pp. 1–6.

[65] 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 *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 979–995.

[66] N. H. Imam and V. G. Vassilakis, "A survey of attacks against twitter spam detectors in an adversarial environment," *Robotics*, vol. 8, no. 3, p. 50, 2019.

[67] 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.

[68] 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.

[69] E. Grolman, H. Binyamini, A. Shabtai, Y. Elovici, I. Morikawa, and T. Shimizu, "Hateversarial: Adversarial attack against hate speech detection algorithms on twitter," in *Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization*, 2022, pp. 143–152.

[70] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, "The rise of social bots," *Communications of the ACM*, vol. 59, no. 7, pp. 96–104, 2016.

[71] 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.

[72] 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 *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, M. Bailey and R. Greenstadt, Eds. USENIX Association, 2021, pp. 4097–4114. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/xu-teng

[73] CujoAI, "Machine learning security evasion competition 2021," https://cujo.com/machine-learning-security-evasion-competition-2021-calls-for-researchers-and-practitioners/, 5 2021.

[74] ——, "Mlsec 2021 anti-phishing evasion track: Task, results, and winning solutions," https://cujo.com/mlsec-2021-anti-phishing/, 10 2021.

[75] Kaspersky, "Mlsec 2021 anti-phishing evasion track: 1st place solution description," https://gist.github.com/KLBot/eaff9a1df3402da709a38cfc83680f0e, 10 2021.

[76] R. Reeves, "Mlsec 2021 anti-phishing evasion track: 3rd place solution description," https://github.com/reevesrs24/mlsec2021, 10 2021.

[77] J. Chen, M. I. Jordan, and M. J. Wainwright, "Hopskipjumpattack: A query-efficient decision-based attack," in *2020 ieee symposium on security and privacy (sp)*. IEEE, 2020, pp. 1277–1294.

[78] Code for our mlsec analysis. [Online]. Available: https://1drv.ms/u/s!AlDgq87-rJINgo4ge3lNRgElmbv_mg?e=0SffrA

[79] (2021) Submission history of mlsec. [Online]. Available: https://1drv.ms/u/s!AlDgq87-rJINgo4ikRuSBbR22FOKGA?e=2sfTJ5

[80] "AI incident database," https://incidentdatabase.ai/, accessed: 2022-09-01.

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

[82] 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 *International Conference on Machine Learning*. PMLR, 2021, pp. 7721–7735.

[83] 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.

[84] S. Shan, A. N. Bhagoji, H. Zheng, and B. Y. Zhao, "Poison forensics: Traceback of data poisoning attacks in neural networks," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 3575–3592.

[85] 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 CCS*, 2022.

[86] Evasive phishing screenshots. [Online]. Available: https://1drv.ms/u/s!AlDgq87-rJINgo4kQh_315BPxxM_Ew?e=Kuzmbh

[87] M. Nasr, A. Bahramali, and A. Houmansadr, "Defeating {DNN-Based} traffic analysis systems in {Real-Time} with blind adversarial perturbations," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2705–2722.

[88] 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 Symp.*, 2021.

[89] 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 *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 694–711.

[90] 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 *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 86–107.

[91] S. Hussain, P. Neekhara, S. Dubnov, J. McAuley, and F. Koushanfar, "{WaveGuard}: Understanding and mitigating audio adversarial examples," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2273–2290.

[92] 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 *CCS*, 2021, pp. 516–534.

[93] E. Bagdasaryan and V. Shmatikov, "Blind backdoors in deep learning models," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1505–1521.

[94] D. Tang, X. Wang, H. Tang, and K. Zhang, "Demon in the variant: Statistical analysis of {DNNs} for robust backdoor contamination detection," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1541–1558.

[95] 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 *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2345–2362.

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

[97] L. Song and P. Mittal, "Systematic evaluation of privacy risks of machine learning models," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2615–2632.

[98] 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 *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 739–753.

[99] 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 *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 275–289.

[100] 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 *Proc. ACM SIGSAC Conf. Comp. Commun. Secur.*, 2020, p. 67–83.

[101] F. Suya, J. Chi, D. Evans, and Y. Tian, "Hybrid batch attacks: Finding black-box adversarial examples with limited queries," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1327–1344.

[102] B. Hui, Y. Yang, H. Yuan, P. Burlina, N. Z. Gong, and Y. Cao, "Practical blind membership inference attack via differential comparisons," in *NDSS*, 2021.

[103] 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 *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1989–2004.

[104] 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 *2021 IEEE symposium on security and privacy (SP)*. IEEE, 2021, pp. 730–747.

[105] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, "High accuracy and high fidelity extraction of neural networks," in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 1345–1362.

[106] 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.

[107] J. Schneider and G. Apruzzese, "Concept-based adversarial attacks: Tricking humans and classifiers alike," in *2022 IEEE Security and Privacy Workshops (SPW)*, 2022.

[108] 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 *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 497–514.

[109] L. Batina, S. Bhasin, D. Jap, and S. Picek, "CSINN: Reverse engineering of neural network architectures through electromagnetic side channel," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 515–532.

[110] 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 *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1919–1936.

[111] M. Handley, V. Paxson, and C. Kreibich, "Network intrusion detection: Evasion, traffic normalization, and {End-to-End} protocol semantics," in *10th USENIX Security Symposium (USENIX Security 01)*, 2001.

[112] 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 Secur. Symp.*, 2019.

[113] H. Zhang, Z. Li, G. Li, L. Ma, Y. Liu, and Z. Jin, "Generating adversarial examples for holding robustness of source code processing models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 1169–1176.

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

[115] "Ai risk management framework: Second draft," NIST, Tech. Rep., August 2022.

[116] I. Lee, "Cybersecurity: Risk management framework and investment cost analysis," *Business Horizons*, vol. 64, no. 5, pp. 659–671, 2021.

[117] 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.

[118] N. Kshetri, "Economics of artificial intelligence in cybersecurity," *IEEE IT Professional*, 2021.

[119] (2022) Microsoft open source toolkits. [Online]. Available: https://www.microsoft.com/en-us/research/group/machine-learning-research-group/open-source/

[120] (2022) Meta bug bounty program info. [Online]. Available: https://facebook.com/whitehat

[121] (2022) Youtube researcher program. [Online]. Available: https://research.youtube/

[122] S. W. Dekker, "Just culture: who gets to draw the line?" *Cognition, Technology & Work*, vol. 11, no. 3, pp. 177–185, 2009.

[123] N. Khatri, G. D. Brown, and L. L. Hicks, "From a blame culture to a just culture in health care," *Health care management review*, vol. 34, no. 4, pp. 312–322, 2009.

[124] 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 *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 673–690.

[125] N. Carlini, "A critique of the deepsec platform for security analysis of deep learning models," *arXiv preprint arXiv:1905.07112*, 2019.

[126] 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 *ICML 2021 Workshop on Adversarial Machine Learning*, 2021.

[127] 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.

[128] 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.

[129] 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 (NDSS) Symposium 2020*, 2020.

[130] 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.

[131] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," *NDSS*, 2020.

[132] 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 *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 712–729.

[133] A. Salem, A. Bhattacharya, M. Backes, M. Fritz, and Y. Zhang, "{Updates-Leak}: Data set inference and reconstruction attacks in online learning," in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 1291–1308.

[134] 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.

[135] V. Chandrasekaran, K. Chaudhuri, I. Giacomelli, S. Jha, and S. Yan, "Exploring connections between active learning and model extraction," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1309–1326.

[136] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. K. Nicholas, "Malware detection by eating a whole exe," in *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[137] W. Guo, D. Mu, J. Xu, P. Su, G. Wang, and X. Xing, "Lemna: Explaining deep learning based security applications," in *proceedings*

of the 2018 ACM SIGSAC conference on computer and communications security, 2018, pp. 364–379.

[138] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, "Detecting ai trojans using meta neural analysis," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 103–120.

[139] G. Lovisotto, H. Turner, I. Sluganovic, M. Strohmeier, and I. Martinovic, "{SLAP}: Improving physical adversarial examples with {Short-Lived} adversarial perturbations," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1865–1882.

[140] 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 *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1381–1398.

[141] Y. Chen, S. Wang, W. Jiang, A. Cidon, and S. Jana, "{Cost-Aware} robust tree ensembles for security applications," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2291–2308.

[142] T. Eisenhofer, L. Schönherr, J. Frank, L. Speckemeier, D. Kolossa, and T. Holz, "Dompteur: Taming audio adversarial examples," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2309–2326.

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

[144] Z. Xi, R. Pang, S. Ji, and T. Wang, "Graph backdoor," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1523–1540.

[145] X. Wu, W. Guo, H. Wei, and X. Xing, "Adversarial policy training against deep reinforcement learning," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1883–1900.

[146] R. Sheatsley, B. Hoak, E. Pauley, Y. Beugin, M. J. Weisman, and P. McDaniel, "On the robustness of domain constraints," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 495–515.

[147] 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 *NDSS*, 2019.

[148] (2022) Yolov3: Real-time object detection algorithm (guide). [Online]. Available: https://viso.ai/deep-learning/yolov3-overview/

[149] Z. Li and Y. Zhang, "Membership leakage in label-only exposures," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 880–895.

[150] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "Textbugger: Generating adversarial text against real-world applications," in *NDSS*, 2019.

[151] 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 *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 85–99.

[152] 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 *NDSS*, 2019.

[153] S. Ma and Y. Liu, "Nic: Detecting adversarial samples with neural network invariant checking," in *Proceedings of the 26th Network and Distributed System Security Symposium (NDSS 2019)*, 2019.

[154] 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 *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 707–723.

[155] E. Quiring, A. Maier, and K. Rieck, "Misleading authorship attribution of source code using adversarial learning," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 479–496.

[156] L. Song, R. Shokri, and P. Mittal, "Privacy risks of securing machine learning models against adversarial examples," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 241–257.

[157] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, "Memguard: Defending against black-box membership inference attacks via adversarial examples," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 259–274.

[158] 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 *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1265–1282.

[159] T. Baluta, S. Shen, S. Shinde, K. S. Meel, and P. Saxena, "Quantitative verification of neural networks and its security applications," in *Proc. ACM SIGSAC Conf. Comp. Commun. Secur.*, 2019, pp. 1249–1264.

[160] B. Wang and N. Z. Gong, "Attacking graph-based classification via manipulating the graph structure," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2023–2040.

[161] Z. Yang, J. Zhang, E.-C. Chang, and Z. Liang, "Neural network inversion in adversarial setting via background knowledge alignment," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 225–240.

[162] R. Schuster, T. Schuster, Y. Meri, and V. Shmatikov, "Humpty dumpty: Controlling word meanings via corpus poisoning," in *2020 IEEE symposium on security and privacy (SP)*. IEEE, 2020, pp. 1295–1313.

[163] E. Quiring, D. Klein, D. Arp, M. Johns, and K. Rieck, "Adversarial preprocessing: Understanding and preventing {Image-Scaling} attacks in machine learning," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1363–1380.

[164] K. Leino and M. Fredrikson, "Stolen memories: Leveraging model memorization for calibrated {White-Box} membership inference," in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 1605–1622.

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

[166] Z. Li, Y. Wu, J. Liu, Y. Chen, and B. Yuan, "Advpulse: Universal, synchronization-free, and targeted audio adversarial attacks via subsecond perturbations," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1121–1134.

[167] S. Abdelnabi, K. Krombholz, and M. Fritz, "Visualphishnet: Zero-day phishing website detection by visual similarity," in *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, 2020, pp. 1681–1698.

[168] Y. Li, M. Li, B. Luo, Y. Tian, and Q. Xu, "Deepdyve: Dynamic verification for deep neural networks," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 101–112.

[169] J. Lin, L. Xu, Y. Liu, and X. Zhang, "Composite backdoor attack for deep neural network by mixing existing benign features," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 113–131.

[170] D. Chen, N. Yu, Y. Zhang, and M. Fritz, "Gan-leaks: A taxonomy of membership inference attacks against generative models," in *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, 2020, pp. 343–362.

[171] H. Huang, J. Mu, N. Z. Gong, Q. Li, B. Liu, and M. Xu, "Data poisoning attacks to deep learning based recommender systems," in *NDSS*, 2021.

[172] S. Abdelnabi and M. Fritz, "Adversarial watermarking transformer: Towards tracing text provenance with data hiding," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 121–140.

[173] X. He, J. Jia, M. Backes, N. Z. Gong, and Y. Zhang, "Stealing links from graph neural networks," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2669–2686.

[174] G. Severi, J. Meyer, S. Coull, and A. Oprea, "{Explanation-Guided} backdoor poisoning attacks against malware classifiers," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1487–1504.

[175] R. Schuster, C. Song, E. Tromer, and V. Shmatikov, "You autocomplete me: Poisoning vulnerabilities in neural code completion," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1559–1575.

[176] J. R. S. Vicarte, G. Wang, and C. W. Fletcher, "{Double-Cross} attacks: Subverting active learning systems," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1593–1610.

[177] 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 *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1901–1918.

[178] H. Jia, C. A. Choquette-Choo, V. Chandrasekaran, and N. Papernot, "Entangled watermarks as a defense against model extraction," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1937–1954.

[179] Y. Zhu, Y. Cheng, H. Zhou, and Y. Lu, "Hermes attack: Steal {DNN} models with lossless inference accuracy," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021.
[180] 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 *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2237–2254.
[181] 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 *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 3793–3810.
[182] 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 *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2255–2272.
[183] 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 *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 108–125.
[184] A. Bahramali, M. Nasr, A. Houmansadr, D. Goeckel, and D. Towsley, "Robust adversarial attacks against dnn-based wireless communication systems," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 126–140.
[185] 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 *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 535–557.
[186] X. He and Y. Zhang, "Quantifying and mitigating privacy risks of contrastive learning," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 845–863.
[187] M. Chen, Z. Zhang, T. Wang, M. Backes, M. Humbert, and Y. Zhang, "When machine unlearning jeopardizes privacy," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 896–911.

Appendix A
Additional Case Studies and Considerations

A. Evasion via Adversarial Examples (and Perturbations)

We provide the formal definition of an evasion attack by means of adversarial examples. We then provide a consideration on the specific *perturbation* that is typically used to craft an adversarial example.

Definition. Let M be a (trained) ML model; and let x be any input sample, corresponding to a given ground truth y; let us assume that such x is correctly analyzed by M, i.e., $M(x)=y$. An adversarial example is defined by means of an "adversarial perturbation" that, when applied to x, will yield the adversarial example $x' = x+\varepsilon$. Such ε corresponds to the perturbation, which must be chosen so that: (1) the ground truth of both the original and adversarial example is the same (i.e., both x and x' correspond to the same y); and (2) the targeted ML model M will output an incorrect label—i.e., $M(x')=y' \neq y$. *Finding* a suitable perturbation can be expressed as an optimization problem (subject to some constraint). Such notation describes an *evasion* attack against ML (a missclassification at test-time).

Consideration. In related literature, the "adversarial perturbation" (i.e., ε) used to craft an adversarial example is typically very small, i.e., imperceptible by humans [39]. However, as pointed out by several works [13], [60], this constraint does not hold universally. Some attackers may want adversarial examples to be *noticed* by humans [107]), while in other domains (e.g., malware [47]) humans do not inspect inputs to the ML model, meaning that perturbations can be of higher magnitude—as long as they are physically realizable [112].

B. Case Study: Malware Competition (2020)

We extend our 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 (i) hardened, but also (ii) 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. To be accepted, such solutions had to meet some performance metrics—at most 1% false positive rate (fpr), 90% true positive rate (tpr), and 5s response time—measured on an *unknown* test set. Accepted solutions will be used for the attack phase. 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 PE malware samples, which had to be manipulated 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 PE 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. [127] 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 who would query the detector—i.e., what would be done by the participants of the second challenge. The solution of [127] was evaded only by 23% of the PE submitted during the attacker's challenge.
- (Attack) Ceschin et al. [128] won the attackers challenge by having all their PE to evade *all* the considered detectors—including the one by Quiring et al. [127]. Notably, the strategy adopted by [128] was *simple*. Specifically they show that "embedding the malware payload into another binary eliminates most detection capabilities presented by the models"

Indeed, to quote from [128]: "[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.

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

We observe that the gain of an attacker may not correspond to the loss of a defender. Consider the following scenario: a company, $\mathcal{D}$, whose main business is the development of ML-based malware detectors. Such a company 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 amounting to $\mathbb{R}$. Two scenarios can occur.

- **Pay.** If $\mathcal{C}$ pays the ransom, then $\mathcal{C}$ loses $\mathbb{R}$ while $\mathcal{A}$ gains $\mathbb{R}$. However, $\mathcal{D}$ (which is the developer of the 'evaded' detector) loses very little. In most cases, the service level agreement (SLA) between $\mathcal{D}$ and $\mathcal{C}$ protect $\mathcal{D}$ in cases of 'faults' in their systems. In contrast, if the SLA determine that $\mathcal{D}$ is liable of 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 if the attack was targeted, such as 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 SLA: if $\mathcal{D}$ is liable, then they must help $\mathcal{C}$ to recover their data; hence, $\mathcal{D}$ may either incur a loss (i.e., human labour), or not (i.e., if they are insured). On the other hand, if $\mathcal{D}$ is not liable, then $\mathcal{D}$ may *gain* something: this may happen if $\mathcal{C}$ asks $\mathcal{D}$ to assist in the recovery of their systems (under a payment).

Simply put, when looking at cybersecurity from the economical (and operational) viewpoint, there are many nuances that can determine the deployment of a countermeasure in an ML system. Such considerations further complicate the decision of whether to develop a countermeasure or not—whose main goal is to be beneficial for the system designer (or their employer).

D. 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—and 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 to avoid giving attackers complete vision of an ML systems or a portion thereof. When constraints of this nature do not exist, researchers and practitioners must provide both the descriptive details and the technical artifacts to 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 [60], 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—which are not part of the contribution of a paper, but which are crucial to provide at least some form of reproducibility.

APPENDIX B
STATE-OF-THE-ART: LITERATURE REVIEW

In this appendix, we will exhaustively describe our review of the SotA (summarized in §IV). Given the detailed methodology, the original interpretations, and the comprehensive summary of recent literature, we consider this appendix as a complementary contribution of our main paper.

A. Methodology

Inspired by by [14] and by [81], we carry out our review by adopting a structured approach organized two phases: *selection* and *inspection*. Both phases entailed exchanges of opinions among the authors of this paper, aimed at ensuring fairness in the review process and removing potential sources of bias.

First phase: Selection. We conduct our survey between July and August 2022. During such timeframe, we performed two following steps: we *get* the proceedings and *filter* them.

i) *Get.* We downloaded the proceedings of the 'top4' cybersecurity conferences (SEC, NDSS, CCS, SP). We considered the editions of these conferences held in the last 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 Workshop publications). Overall, these proceedings amounted to 435 papers for 2019, 470 for 2020, and 644 for 2021.

ii) *Reduce.* 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[8]. At the end of this step, we obtain 30 papers for 2019, 35 papers for 2020, and 66 papers for 2021.

Second phase: Inspection. We then further analyze our set of 131 candidate papers, with a twofold objective: *filtering* those papers that, despite being related to ML security, were beyond our scope; and *distilling* the knowledge that we want 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 that we excluded from our main analysis due to assumptions that significantly deviate from our main focus. Specifically: *federated learning* [130], because the corresponding issues mostly pertain to the distributed systems domain (such as byzantine fault tolerance [131]); papers considering *robust, efficient, or secure computation*, because they mostly focus on technical implementations and do not propose or consider any adversarial ML attack; as well as those papers that provide *high-level analyses*, because they do not propose any original attack or defense in the ML security context. At the end of such filtering, we obtain 23 papers for 2019, 24 papers for 2020, and 41 papers for 2021.

ii) *Distill.* Finally, we thoroughly analyze the final set of 88 papers, We do this by going through each paper and asking ourselves a set of 12 questions—each focused on deriving knowledge significant for our 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 carry out our analysis in pairs: two authors independently reviewed each paper and then discussed the findings during a series of meetings. Such process is repeated a total of five times.

Before we delve deeper into our analysis, we provide some interesting trivia that we noticed while scanning the proceedings. In 2019, most papers were put in sessions entirely devoted to "ML security". In 2021, however, we had sections entirely dedicated to specific attacks (e.g., the "attacks on speech recognition", or "inference" at SP2021). Some papers have very similar contributions across the same conference. For instance, [89] and [132] (both at SP21) propose attacks on speech recognition (and both, surprisingly, evaluate their proposals also on real humans); whereas [133] (SEC20) and [22] (CCS20) propose membership inference attacks that leverage the updates of ML models.

[8]For instance, the well-known work by Aghakhani et al. [129] is included in the "Malware 2" section of NDSS20, but it (also) considers attacks against ML-based malware detectors

B. Research Questions

We then 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). Specifically, 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)?
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) are more detailed and focus on the threat model envisioned in the paper. Specifically, we focus our attention at the *weakest*[9] attacker considered in any paper, and then ask ourselves:

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

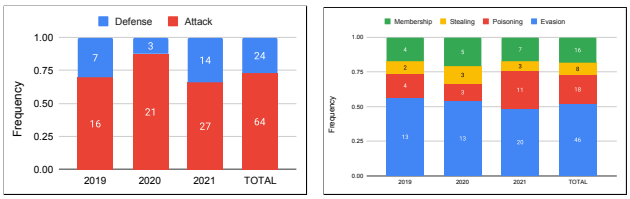
We explain how we answered T1 to T4 in Appendix B-H.

A summary of all the answers to all our questions for all our considered papers is provided in Table IV, which is 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.

[9]We explain our reason in Appendix B-I.



(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 relatively 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 [134]) but also evaluated such 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., [52]) also discuss some countermeasures to the corresponding attack. Such a result is not surprising, as 'attack papers' are typically considered to have a higher novelty value—mostly due to the fact that 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 such a 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.

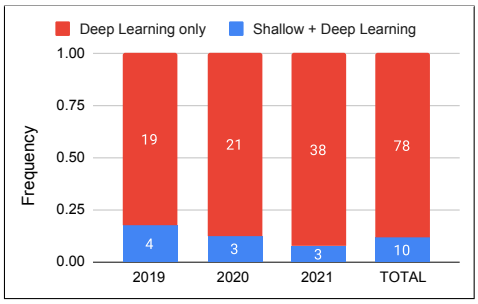


Fig. 8: G3: what is the considered ML *paradigm*?

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 algorithms (interestingly, two papers consider *only* SL: [47] and [88]). We believe that future efforts should put more focus on SL, for a twofold reason:

- *in some domains, SL is better than DL* [19], [129], [135]. For instance, we quote from [129]:"Even though in our experiments we used SVM, deep neural networks (i.e., MalConv [136]), 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 [137]."
- *SL exhibit different properties than DL.* We find insightful to quote a statement by Xu et al. [138] (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 such 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.

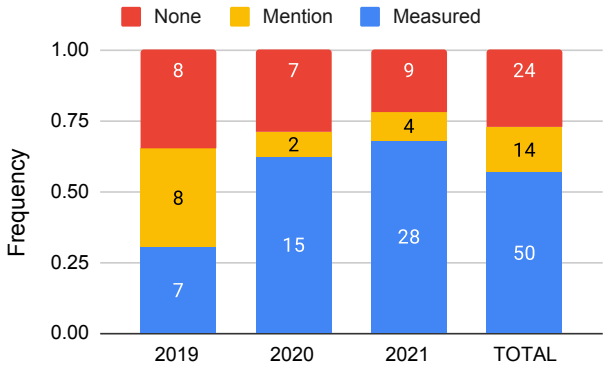


Fig. 9: G4: are the *costs* taken into account (in *any* way)?.

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. Such occurrence 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 inspiring papers which attempted to *quantify* the cost of an attack in real currency.

- (Equipment expenditure) Lovisotto et al. [139] attempt to evade object detectors via artificial light manipulation by means of *commercial projectors*. Some of these projectors have a MSRP of few hundreds dollars, whereas others can be much more expensive.
- (Queries expenditure) The authors of [32] and [135] 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 such 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 by considering the 'tradeoff', i.e., the change in performance *after* applying a given countermeasure (e.g., [140]). Let us mention two notable 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. [141] propose to measure the cost of feature manipulation against shallow ML models. The intuition is to carry out such measurements during the *evaluation* of ML models, with the underlying principle of "if an attacker wants to evade such 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 [141] does *not* evaluate any attack[10]: the main contribution is measuring the (computational) effort in manipulating some features.
- Heisenhofer et al. [142] acknowledge that attackers can use adversarial examples to evade a ML system for automated speech recognition. Hence, [142] aims to increase the cost of the attacker to physically realize 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 such detection can occur at different *granularities*. For instance, some papers aim to detect individual adversarial examples (e.g., [91]) or poisoned ML models (e.g., [94]); whereas others have a broader scope and aim to detect attackers (e.g., [100]).

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* [143] 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., [97]) consider multiple domains, which is why the y-axis goes above 1. More details are shown in Table IV.

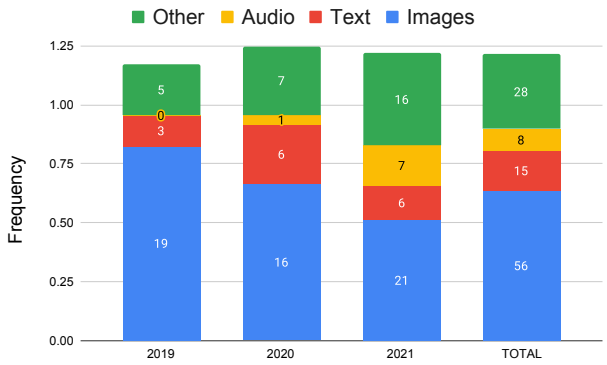


Fig. 10: What are the *data-types* considered in the evaluation?

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 such 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 [144] or games [145].

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.

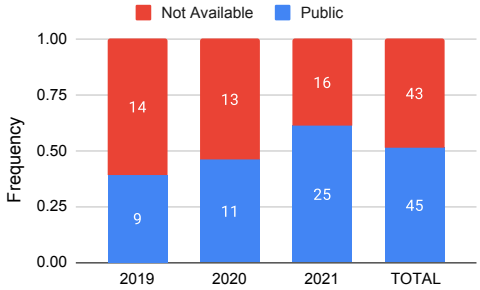


Fig. 11: Has the *source-code* been publicly released?

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 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, in [52] the paper 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 [134] 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 are meant to pinpoint the degree of 'realism' considered in the evaluations carried out by prior

[10]Which is why we do *not* include [141] in our Table IV.

work. Answering these two questions was not simple. Indeed, 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 such 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 such samples do not violate domain constraints [146]). Simply put, to answer G7 we embraced Abdullah et al. [132] 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." The answers to G7 are shown in Fig. 12.

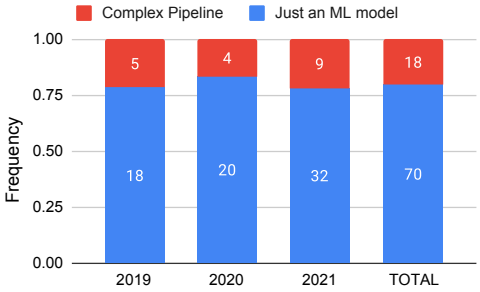


Fig. 12: Has a complex *pipeline* been reproduced in the evaluation?

As expected, nearly 80% of papers perform their attacks by considering a single ML model. Even some papers that attack MLaaS (e.g., [147]) ultimately simply have the MLaaS train a given ML model, and then attempt to violate such model without reproducing a real pipeline. Nonetheless, there are some notable efforts, such as [47] and [20]. These works reproduce a custom ML system by developing *also* a preprocessing component: such 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., [23]) automatically fall into this category; however, we also considered papers that used 'extremely popular' ML models (e.g., Yolo [148]) 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 [47]. 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 the INVISIBLE ones). The answer to the second part of G8 can be seen in the detailed Table IV.

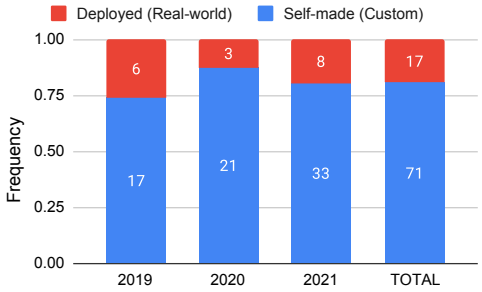


Fig. 13: Does the paper consider an ML model *deployed* in the real world?

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. [23] 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. [23], but it must be acknowledged that similar experiments are "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 statement (§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

These questions were also particularly difficult to answer. Indeed, we found that there is a remarkable lack of clarity in the way that research papers present their envisioned threat models (as also hinted in §IV-D).

T1: Parameters. This question 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, the optimizer, 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. [94], 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. For *semantics*, we consider whether the attacker was aware of what was fed as input to 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 such information. An example is [108], in which the attacker simply does not need such 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. [105], 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., [149]), if the attacker received the class with the highest probability; *probability*, if the attacker received any additional information beyond the label (e.g., [103]); as well as *none*, when the attacker did not receive any feedback (or was not needed).

T4: Training. As also stated in our main paper (see §IV-D), identifying if the attacker had some knowledge on the training data was also very confusing. We attempted to be as specific as possible by considering the following answers: *full*, if the attacker had full access (both read and write) to the training data; *read*, if the attacker can observe the entire training set; *subset* (e.g., [50], if the attacker had a subset of the actual training dataset; *surrogate* (e.g., [133]), if the attacker had samples not included in the actual training data, but having the same distribution; *distribution* (e.g., [150]), if the attacker only knows the class distribution of the training data; and *none* if the attacker knows nothing.

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). Nevertheless, the individual answers to these questions are provided in Table IV, described in the next appendix.

I. Complete Table

We summarize our complete literature review in Table B-I. This table 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. In particular:

- *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., [129]). 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 [108]).
- *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 [93]).

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., [42] and [151]). 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.

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 [147]	atk	Member.	DL	◐	✓	✓			✓		CLOSED	✗	✓	p	✗
		Li [152]	atk	Evasion	DL	✓	✓				✓			✓	✓	✗	R
		Ma [153]	def	Evasion	DL	◐	✓							✓	✓	p	✗
		Li [150]	atk	Evasion	DL+SL	◐		✓				✓	CLOSED	✗	✓	p	D
	SP (3/84)	Ling [124]	def	Evasion	DL	✓	✓				✓			✓	✓	p	✗
		Wang [154]	def	Poison.	DL	✓	✓				✓			✓	✓	✗	✓
		Nasr [98]	atk	Member.	DL	✗	✓			Finance				✗	✓	p	D
	SEC (6/113)	Tong [112]	def	Evasion	DL+SL	✓				Malware	✓			✗	✓	p	✗
		Demontis [42]	atk	Evasion	DL+SL	✗	✓			Malware				✗	✓	p	S
		Xiao [20]	atk	Evasion	DL	✗	✓					✓	CLOSED	✗	✓	p	✗
		Quiring [155]	atk	Evasion	DL+SL	✗		✓						✗	✓	p	✗
		Hong [108]	atk	Evasion	DL	✗	✓					*		✗	✗	p	✗
		Batina [109]	atk	Stealing	DL	◐	✓					*		✗	✓	p	✗
	CCS (10/149)	Song [156]	atk	Member.	DL	✗	✓				✓			✗	✓	p	✗
		Jia [157]	def	Member.	DL	✗	✓			+				✗	✓	p	✗
		Co [99]	atk	Evasion	DL	✓	✓							✗	✓	p	✗
		Liu [158]	def	Poison.	DL	✓	✓							✓	✓	p	✓
		Baluta [159]	def	Poison.	DL	◐	✓				✓			✓	✓	p	✓
		Zhao [103]	atk	Evasion	DL	✗	✓				✓		OPEN	✗	✓	p	S
		Tramer [12]	atk	Evasion	DL	◐	✓				✓	✓	CLOSED	✗	✓	s	✗
		Wang [160]	atk	Evasion	DL	✓				Graphs				✗	✓	✗	⊂
		Yao [50]	atk	Poison.	DL	◐	✓							◐	✓	p	⊂
		Yang [161]	atk	Stealing	DL	◐	✓						CLOSED	✗	✓	p	D
2020 (24/470)	NDSS (2/88)	Aghakhani [129]	atk	Evasion*	DL+SL	✗				Malware	✓			✗	✓	✗	D
		Yu [32]	atk	Stealing	DL	✓	✓						CLOSED	✗	✓	p	D
	SP (4/104)	Schuster [162]	atk	Poison.	DL	✓		✓						✗	✓	✗	⊂
		Pierazzi [47]	atk	Evasion	SL	✓				Malware	✓	✓		✓	✓	p	R
		Chen [77]	def	Evasion	DL	✓	✓				✓			✗	✓	l	✗
		Jan [134]	atk	Evasion*	DL	✓				Network				✗	✓	p	⊂
	SEC (8/157)	Salem [133]	atk	Member.	DL	✗	✓			Location				◐	✓	p	S
		Chandrasekaran [135]	atk	Stealing	DL+SL	✓	✓	✓		+				◐	✓	p	✗
		Suya [101]	atk	Evasion	DL	✓	✓				✓			✗	✓	p	S
		Jagielski [105]	atk	Stealing	DL	◐	✓							✗	✓	l	D
		Quiring [163]	atk	Evasion	DL	✗	✓				✓	✓		✗	✓	p	✗
		Li [140]	def	Evasion	DL	✓		✓						✗	✓	p	D
		Leino [164]	atk	Member.	DL	✓	✓			+				✓	✓	p	✗
		Zhang [165]	atk	Evasion	DL	✗	✓							✓	✓	p	✗
	CCS (10/121)	Nassi [23]	atk	Evasion	DL	✗	✓				✓	✓	CLOSED	✗	✓	s	✗
		Li [166]	atk	Evasion	DL	✗			✓					✓	✓	p	⊂
		Shan [100]	def	Evasion	DL	◐	✓				✓			✓	✓	p	✗
		Pang [151]	atk	Poison.	DL	✓	✓				✓			✓	✓	p	✓
		Abdelnabi [167]	atk	Evasion*	DL	✗	✓			Phishing	✓	✓		✗	✓	✗	✗
		Li [168]	atk	Evasion	DL	✓	✓							✓	✓	p	✗
		Lin [169]	atk	Poison.	DL	✓	✓	✓			✓			✓	✓	✗	✓
		Chen [170]	atk	Member.	DL	✓	✓							✗	✓	p	D
		Zanella [22]	atk	Member.	DL	✓		✓			✓			✗	✓	p	✗
		Song [21]	atk	Member.	DL	✓		✓						✗	✓	p	S
2021 (41/644)	NDSS (3/87)	Hui [102]	atk	Member.	DL	✗	✓			+	✓			✗	✓	p	✗
		Huang [171]	atk	Poison.	DL	✓				Ratings				✗	✓	✗	⊂
		Barradas [88]	atk	Evasion*	SL	✓				Network	✓			✗	✗	✗	✗
	SP (5/115)	Xu [138]	def	Poison.	DL	✓	✓		✓		✓			✓	✓	p	✓
		Abdelnabi [172]	atk	Evasion	DL	◐		✓			✓	✓		✗	✓	p	✗
		Chen [89]	atk	Evasion	DL	✓			✓		✓	✓	BOTH	✗	✓	s	S
		Abdullah [132]	atk	Evasion	DL	✓			✓			✓	CLOSED	✗	✓	✗	✗
		Nasr [33]	def	Evasion	DL	✓	✓			Finance				✗	✓	p	✗
	SEC (24/246)	Sato [26]	atk	Evasion	DL	✓	✓				✓	✓	OPEN	✓	✓	p	✗
		Nasr [87]	atk	Evasion	DL	✓				Network	✓	✓		✗	✓	p	D
		He [173]	atk	Member.	DL	◐				Graph				✗	✓	p	✗
		Severi [174]	atk	Poison.	DL+SL	✓				Malware	✓			✗	✓	p	✓
		Bagdasaryan [93]	atk	Poison.	DL	✓	✓	✓			✓			✗	✗	✗	✗
		Xi [144]	atk	Poison.	DL	✓				Graph	✓			◐	✓	✗	S
		Tang [94]	def	Poison.	DL	✗	✓				✓			◐	✓	p	⊂
		Schuster [175]	atk	Poison.	DL	✓		✓					OPEN	✗	✓	✗	⊂
		Carlini [52]	atk	Poison.	DL	✓	✓							✗	✓	✗	⊂
		Vicarte [176]	atk	Poison.	DL	✓	✓							✗	✓	p	✗
		Lovisotto [139]	atk	Evasion	DL	✓	✓				✓		OPEN	✗	✓	✗	⊂
		Carlini [28]	atk	Member.	DL	✓		✓				✓	OPEN	✗	✓	p	✗
		Han [95]	atk	Evasion*	DL	✗				Graph				✗	✓	p	✗
		Eisenhofer [142]	def	Evasion	DL	✓			✓		✓	✓	OPEN	✓	✓	p	R
		Wu [145]	atk	Poison.	DL	✓				Games	✓			✗	✓	s	✗
		He [177]	atk	Stealing	DL	✓	✓							✗	✓	l	S
		Rakin [110]	atk	Evasion	DL	✓	✓				✓	*		✗	✓	p	✗
		Jia [178]	def	Stealing	DL	✓	✓		✓		✓			◐	✓	l	⊂
		Zhu [179]	def	Stealing	DL	✗	✓					*		✗	✓	p	✗
		Xiang [180]	def	Evasion	DL	✓	✓				✓			✓	✓	p	✗
		Lin [181]	atk	Evasion *	DL	✓	✓			Phishing	✓	✓		✗	✓	p	✗
		Azizi [182]	def	Poison.	DL	✗		✓			✓			✓	✓	p	✓
		Hussain [91]	def	Evasion	DL	✗			✓		✓			✓	✓	p	✗
		Song [97]	def	Member.	DL	✓	✓			+	✓			✗	✓	l	✗
	CCS (9/196)	Zheng [90]	atk	Evasion	DL	◐			✓			✓	CLOSED	✗	✓	✗	✗
		Mu [183]	atk	Evasion	DL	✓				Graphs				✗	✓	l	✗
		Bahramali [184]	atk	Evasion	DL	✓				Network				✗	✓	✗	S
		Sheatsley [146]	atk	Evasion	DL	✗				Network				✓	✓	p	R
		Du [92]	def	Evasion	DL	✓	✓	✓						✓	✓	p	R
		Li [185]	def	Evasion	DL	✓	✓				✓			✗	✓	✗	✗
		He [186]	def	Member.	DL	✗	✓				✓			◐	✓	l	S
		Li [149]	atk	Member.	DL	✓	✓				✓			✗	✓	l	✗
		Chen [187]	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**