

# Security for Machine Learning-based Software Systems: a survey of threats, practices and challenges

HUAMING CHEN, The University of Sydney, Australia

M. ALI BABAR, CREST - The Centre for Research on Engineering Software Technologies, The University of Adelaide, Australia and Cyber Security Cooperative Research Centre, Australia

The rapid development of Machine Learning (ML) has demonstrated superior performance in many areas, such as computer vision, video and speech recognition. It has now been increasingly leveraged in software systems to automate the core tasks. However, how to securely develop the machine learning-based modern software systems (MLBSS) remains a big challenge, for which the insufficient consideration will largely limit its application in safety-critical domains. One concern is that the present MLBSS development tends to be rush, and the latent vulnerabilities and privacy issues exposed to external users and attackers will be largely neglected and hard to be identified. Additionally, machine learning-based software systems exhibit different liabilities towards novel vulnerabilities at different development stages from requirement analysis to system maintenance, due to its inherent limitations from the model and data and the external adversary capabilities. The successful generation of such intelligent systems will thus solicit dedicated efforts jointly from different research areas, i.e., software engineering, system security and machine learning. Most of the recent works regarding the security issues for ML have a strong focus on the data and models, which has brought adversarial attacks into consideration. In this work, we consider that security for machine learning-based software systems may arise from inherent system defects or external adversarial attacks, and the secure development practices should be taken throughout the whole lifecycle. While machine learning has become a new threat domain for existing software engineering practices, there is no such review work covering the topic. Overall, we present a holistic review regarding the security for MLBSS, which covers a systematic understanding from a structure review of three distinct aspects in terms of security threats. Moreover, it provides a thorough state-of-the-practice for MLBSS secure development. Finally, we summarise the literature for system security assurance, and motivate the future research directions with open challenges. We anticipate this work provides sufficient discussion and novel insights to incorporate system security engineering for future exploration.

**CCS Concepts:** • General and reference → Surveys and overviews; • Security and privacy → Software security engineering; • Computing methodologies → Machine learning: Neural networks.

Additional Key Words and Phrases: machine learning, system security, secure development practices, software engineering

## ACM Reference Format:

Huaming Chen and M. Ali Babar. 2023. Security for Machine Learning-based Software Systems: a survey of threats, practices and challenges. 1, 1 (December 2023), 36 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

## 1 INTRODUCTION

**Background of this work:** Machine learning models have dominantly outperformed many traditional statistical algorithms and models in various application areas, such as tasks from computer vision, video, speech and natural language recognition [189, 206], with an enhanced prediction and inference capability. Considering machine learning models as the core, one key research area is to improve the model performance dedicated to the systems both practically

---

Authors' addresses: Huaming Chen, The University of Sydney, Sydney, Australia, huaming.chen@sydney.edu.au; M. Ali Babar, CREST - The Centre for Research on Engineering Software Technologies, The University of Adelaide, Adelaide, Australia, Cyber Security Cooperative Research Centre, Australia, ali.babar@adelaide.edu.au.

---

2023. Manuscript submitted to ACM

and theoretically with advanced functionalities, such as enhancing the object detection capability [16], human activity recognition [25], boosting the recommendation service and facilitating the decision making process [204]. Typically, the success of such a machine learning-based software system requires tremendous efforts and dedicated resources by the people jointly from different research areas, e.g. machine learning, software engineering and data science. Similar to the classic software systems, the secure development of MLBSS is founded by the collaborative work regarding requirement engineering, framework/system/operation/process design, model development, system implementation and maintenance. A substantial interest has been attracted to address the development challenges for machine learning-based software systems, which predominantly focus on the aspects of *technical debts* and *quality assurance*.

**Motivation of this work:** Recent studies have revealed that security threats for MLBSS could result in catastrophic consequences, particularly when it is targeted on safety-critical software systems. In 2020, the Software Engineering Institute at the CERT Coordination Center (SEI/CERT) has released a first documented vulnerability note for one common vulnerability and exposures (CVE) in a machine learning-based commercial software system [163]. Another security-related system example is the supervisory control and data acquisition systems (SCADA), in which the intrusion detection system plays a vital role to protect SCADA from intrusion [171]. Supervised machine learning model is one pivotal component for the detection of intrusion to ensure the SCADA security. While machine learning models exhibit uncertainty and more attacking vectors are available for adversarial attacks, the security of machine learning-based safety-critical software systems becomes a major concern. One trending sub-topic is to secure the ML/DL models with robustness [12]. While recent works have centered on the security and defense methods for deep learning models [55, 136], there will be a cat-and-mouse game.

Thus, how to systematically secure MLBSS in practice is important and now a hot topic for the academia and industry. On the one side, the *state-of-the-practice* has demonstrated a lack of diligence to ensure the deployment of machine learning-based systems following standardized processes, even with software testing techniques [164]. Most of the research focus has been on addressing the challenges in deploying the machine learning models in the practical applications from the software engineering point of view [3, 95]. For example, over 3,000 questions about DL software deployment to server/cloud/mobile/browser platforms from Stack Overflow were evaluated and a taxonomy consisting of 72 categories linked to challenges in the deployment was recently discussed [26]. The issues of efficiency and effectiveness in adopting methods and metrics from tradition software are reported to understand the challenges while lacking the practitioners face [2, 202, 205, 208]. The ethics, the end users' trust and the security issues as the cross-cutting aspects have been raised as general challenges hindering the scalability [135]. To alleviate the latent aspects, recently a framework named Machine Learning Technology Readiness Levels (MLTRL) framework is proposed to ensure the system operating in a robust, reliable and responsible manner [84, 85].

On the other side, enabling the advanced MLBSS with current available practices will expose a even larger vulnerability surface while the consideration of security is not fully blended. Thus, more joint efforts from academia and industrial partners are desired to provide an overall view encompassing the identification of latent vulnerability root causes and the attack consequences understanding, to consolidate the system defence capabilities in MLBSS. One recent example is that, over 87% Android apps based on TensorFlow Lite from Google Play are structurally similar to each other, which has resulted a successful adversarial attack on all 10 representative Android apps [64]. For edge devices, it requires more insights while deploying machine learning models in terms of the transferability of customised attack modes. Another example is from the system design level, for which the attacks will be easy to be propagated and transferred between systems with the convenience of open source ML/DL libraries and pre-trained models from distributed resources.

**Our aim:** In this work, we aim to investigate the security of MLBSS with current secure development practices. System security engineering has been a long-term and critical research topic, whose primary objective is to ‘minimize or contain defence system vulnerabilities to known or postulated security threats and to ensure that developed systems protect against these threats.’ [35]. To achieve the goals of system security engineering, it is demanded to obtain a comprehensive understanding of security threats for MLBSS covering the novel attack modes and affecting development stages [152]. Since the exploitation of vulnerability in MLBSS may occur at any stages, such as the requirement engineering and model development, how to distill the secure development practices is utterly important. For example, the internal defective design and incomplete secure practices for MLBSS could lead to the system failure by exposing the latent vulnerability to external dedicated attackers [164]. This happens when the practitioners could not transform the problem into requirement in an accurate and secure manner, which may bring undesirable implementation framework and process. Moreover, the vulnerability underlying the machine learning algorithms are considered as inherent limitations that could not be totally fixed yet. Thus, the external adversary with a dedicated level of machine learning knowledge will have the chance to attack the system either by poisoning malicious data or by the existing vulnerability transmitted from other ML models.

To address this question, utilising the system security tools, including the threat model and CIA (confidentiality, integrity and availability) triad, for the integrated analysis of such system, will be firstly explored. In this work, one consolidated contribution is that we obtain the insights towards leveraging secure development practices for MLBSS at different software development stages for the emerging various attack modes. In details, we focus on the following research questions:

- RQ1: What are the security threats in machine learning-based software systems? (Section 3)
- RQ2: What are the state-of-the-art secure development practices in machine learning-based software systems? (Section 4)
- RQ3: What are the challenges and future directions for current practices in machine learning-based software systems? (Section 5)

## 2 EXISTING WORKS AND THE SURVEY STRUCTURE

### 2.1 Existing works

It is inspiring to witness the prominent research trend to utilising artificial intelligence techniques, particularly machine learning, in modern software systems. This brings an increasing number of surveys that summarise the works from different perspectives with the interest of a specific application domain, such as addressing the cloud computing security threats with different types of machine learning methods [21, 46, 126]. A vast majority of these surveys has limited the scope either on a particular topic of the software engineering or a typical type of machine learning, which we have collectively depicted in Fig. 1. Overall, there are three general categories of these works, which are ‘assurance’, ‘software engineering practice’ and ‘security&privacy’. Table. 1 lists the comparison between our work and the existing work.

The first group of existing review is related to the ‘assurance’ of software systems. An exploring study for the quality assurance of deep learning based system was conducted by Liu et al. [95], initialising a discussion limited to deep learning based system engineering. While secure deep learning engineering is presented as a future approach for quality assurance, actionable items are not identified for the relevant process. An instant activity is referred to testing, one dominant methodology to capture the model behavior from a small group of studies. Zhang et al. [202] has thus surveyed the topic of machine learning testing to understand the different scenario for machine learning-based

software systems. Recently, a comprehensive survey for the assurance of machine learning, i.e., its lifecycle and the generation of assurance evidence, is presented by Ashmore et al. [3]. It has generally discussed the machine learning errors. Meanwhile, the methods which provide evidence to support ML components fitting for the purpose and could be adequately integrated into the systems are also included. The key point is raised that the assurance should cover all stages of ML lifecycle. Another recent work by Felderer et al. [40] has highlighted the challenges on quality assurance encountered in the intersection between software engineering and artificial intelligence for artificial intelligence-based systems, which has exhibited a control transfer from typical source code to data.

Another group of existing survey has a focus on the ‘software engineering practices’ (SE practices) for machine learning-based software systems. The work by Martínez-Fernández et al. [110] has explicitly discussed the definition of artificial intelligence-based systems, which core functionalities are enabled by at least one AI component (e.g., for image- and speech-recognition, and autonomous driving). It targets the topic of ‘software engineering for artificial intelligence’ (SE4AI) by reviewing the works that explore software engineering practices to develop, maintain and evolve AI-based systems. Another work by Lwakatare et al. [102] is a review work that investigated the well-established development guidelines for traditional software systems in the scenario of industrial settings to develop and maintain the large-scale machine learning-based software systems. Specifically, there is a plethora of works investigating the deployment process of machine learning-based software systems. Chen et al. [26] has conducted a survey that mined and analysed 3,023 posts from public developer forums to identify the encountered specific challenges in the process of deep learning-based software systems deployment. Other works have either utilised systematic literature review method [48, 73, 135] or conducted interviews with developers [33] to understand the challenges and solutions in deploying machine learning-based software systems.

The last group targets at ‘security&privacy’. Begga et al. [6] and Braun et al. [19] are the earliest works that anticipate to understand the security and privacy threats in intelligent systems, in which they focus on mobile agent-based applications and smart cities, respectively. While it is a general review, Papernot et al. [136] initialised the discussion of security and privacy issues for machine learning-based software systems by exploring the threat model spaces for the learned models. He et al. [55] is so far the first work that review the potential attack modes towards deep

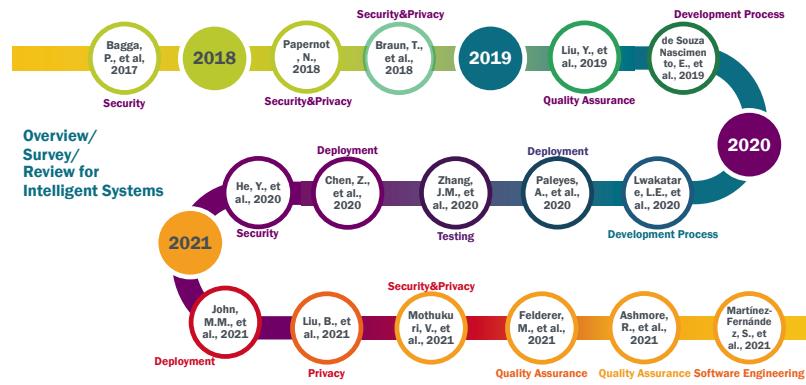


Fig. 1. Existing work of overview/survey/review for intelligent systems.

learning-based systems, in which four types of attacks were summarised. Privacy preservation problem for machine learning is surveyed by Liu et al. [92]. Mothukuri et al. [119] presented that the security and privacy issues persist and will demand further investigation and collaboration for federated machine learning. However, these works have limited the discussion to survey security threats from the algorithmic perspective.

## 2.2 Comparison with other works

While it is similar to the development of traditional Internet infrastructure and system, the machine learning-based software systems are more complicated by the fact that it currently lacks in-depth understanding of the decision-making process. In Table. 1, a comparison with existing works is presented. It is noted that, with the definition of SE practices, some works have focused on the discussion of software quality assurance [40, 95]. A particular interest of providing evidence as assurance to support MLBSS is largely surveyed in [3]. While these works are dedicated to assuring the MLBSS behaviour, the fundamental security development practice is missing. In the work of [202], machine learning model testing, as a key method to mitigate the security threats, is broadly discussed from the perspective of testing process. Other works incorporating the SE practice, such as [73, 102, 110], have missed the discussion of security threats in MLBSS. For the rest works, they have demonstrated a similar research topic related to the security. Some have identified the distinct attack types and privacy issues [6, 19, 55, 92], such as adversarial attack [22, 173], poisoning attack [49], and membership inference attack [62] and so on. In summary, a holistic understanding of security threats for the emerging various attacks is missing.

A most related work by [140] has been published in 2018 to discuss the attacks and defences of ML systems. It is motivated to see the highlights of CIA model for attack surfaces characteristics and the consideration of machine learning pipeline for adversarial goal identification. While it has aimed at a theoretical understanding of the sensitivity of machine learning models, it has successfully attracted a huge amount of research to the machine learning attacks and defences. However, the scope is limited to the structural elements of *machine learning algorithms* and the *data used to train them*, which has now been outdated. Overall, there is no literature review that is aimed at a comprehensive investigation of machine learning-based software systems from the secure development aspect, particularly with the understanding of systematic analysis of security threats and security engineering practices.

Since the interference of such intelligent systems could occur at many stages in the virtual and physical world [57], there is a need to thoroughly investigate the secure development practices for the machine learning-based software systems [113, 160] with the lens of system security. Thus, this work will **firstly** contribute an exclusive taxonomy of attack, which also includes the discussion of details for the identified attack vectors, the impacts and potential defence strategies as the **second contribution**. Most importantly, it advocates this work to capture the state-of-the-practices for MLBSS system security as the **third contribution**. **Lastly**, it provides a conjunction discussion of security practices, which will help us to identify the limitations and future research directions.

## 2.3 Methodology

We have implemented this work as a structured literature review, with an inspiration of using broadly defined keyword search string for relevant literature identification. We firstly designed the search strings: “(‘vulnerability’ OR ‘privacy’ OR ‘attack’) AND (‘machine’ OR ‘deep’ OR ‘reinforcement’ OR ‘federated’) AND ‘learning based system’”, “(‘vulnerability’ OR ‘privacy’ OR ‘attack’) AND (‘artificial intelligence system’ )”, “‘security’ AND (‘machine’ OR ‘deep’ OR ‘reinforcement’ OR ‘federated’) AND ‘learning based system’”. We aims to cover as many impactful paper as possible by soliciting the results from the sources from DBLP, Google Scholar, web in general formats such as papers, patents, courses, slides,

| Study                                | Subtopics                                 | Domain                          | SE practice | System security | Secure development practice |
|--------------------------------------|---|---------------------------------|-------------|-----------------|-----------------------------|
| Martínez-Fernández et al. 2021 [110] | SE practices                              | AI-based systems                | ✓           | ✗               | ✗                           |
| Asmore et al. 2021 [3]               | Quality Assurance                         | ML lifecycle                    | ✗           | ✓               | ✗                           |
| Felderer et al. 2021 [40]            | Quality assurance                         | AI-based systems                | ✓           | ✗               | ✗                           |
| Liu et al. 2021 [92]                 | Security&Privacy                          | Machine learning                | ✗           | ✗               | ✗                           |
| He et al. 2020 [55]                  | Security&Privacy                          | Deep learning systems           | ✗           | ✓               | ✗                           |
| John et al. 2020 [73]                | SE practices                              | Edge/cloud/hybrid architectures | ✓           | ✗               | ✗                           |
| Zhang et al. 2020 [202]              | SE practices                              | ML system                       | ✗           | ✓               | ✗                           |
| Lwakatare et al. 2020 [102]          | SE practices                              | Industrial software systems     | ✓           | ✗               | ✗                           |
| Liu et al. 2019 [95]                 | Quality assurance                         | Deep learning systems           | ✗           | ✓               | ✗                           |
| Papernot et al. 2018 [140]           | Security&Privacy                          | ML-based systems                | ✓           | ✓               | ✗                           |
| Braun et al. 2018 [19]               | Security&Privacy                          | Smart cities                    | ✗           | ✓               | ✗                           |
| Bagga et al. 2017 [6]                | Security&Privacy                          | Mobile agent-based applications | ✗           | ✓               | ✗                           |
| Our work                             | Security and secure development practices | ML-based software systems       | ✓           | ✓               | ✓                           |

Table 1. Comparison between our work and the existing work related to intelligent systems

tutorials, reports, articles and so on. After reading the titles and the abstracts, together with the backward and forward snowballing, we have included 141 papers for our survey. Particularly, we have identified 52 paper related to the security threats for MLBSS covering data- (15 papers), model- (30 papers), and system-oriented (7 papers) attacks. We wish not to claim that all the papers in this are collected, especially when this area is emerging in such an exceptional speed. However, we believe that our data collection process has covered most of the key studies to provide the security issues and practice for MLBSS.

#### 2.4 Overall structure of the work

Following, with the collected state-of-the-art literature works, we present the taxonomy of the latest security threats and the corresponding secure development practices in relation to different stages. The sections are organised as follows: section 3 presents a taxonomy of security threats in MLBSS as well as the attack types. Section 4 highlights the state-of-the-practice for MLBSS and the limitations of adopting current practices to secure MLBSS. The open challenges and future directions are presented in section 5 while section 6 concludes the paper.

### 3 ATTACK TAXONOMY FOR SYSTEM SECURITY IN MLBSS

The recent immense success of machine learning-based software systems is not possible without the accumulating big data and cutting edge machine learning techniques [111]. For MLBSS, the advancement of machine learning models, particularly for the deep learning models with over hundred thousands parameters, is the critical contributor to learn the latent deep relationships from the big data to generate the outstanding prediction, inference as well as the reasoning performance in comparison to human intelligence. Moreover, it will not be possible without the development of computational infrastructure. Including the high performance computing cluster and the graphics processing unit (GPU), the supporting infrastructure for MLBSS has now become more complicated by including more advanced computational resources to support the model training and system deployment. These contributing factors are briefly discussed in [119], and the recent representative applications include the AlphaFold 2 for science from DeepMind [74] and various mobile applications with on-device model for Android applications [64]. These factors have posed novel security threats, and new attack surfaces can be leveraged by attackers in either a proactive or reactive manner.

The security threats in empowered software systems have recently aroused attention from different areas, including the general AI and cybersecurity communities. Nonetheless, a dominant focus has been limited to the aspects of algorithms and data. Via the systematic analysis of machine learning-based software systems, we first discuss the system security threats with confidentiality, availability and integrity (CIA) triad from the perspective of information security practitioners and researchers [9, 139, 157].

1. Confidentiality: it indicates the sensitive information and private data are not available or disclosed to unauthorised access, including individuals, entities and processes.

2. Integrity: it means the completeness and trustworthiness of data over the entire data lifecycle in system.

3. Availability: it guarantees that the information service are always online and available as intended at all times.

While CIA triad could serve as good metrics for analyzing the security issues for MLBSS, it is originally designed for a general course of security analysis. For example, the confidentiality for traditional software systems could be digital data compromised, leakage and theft such as hardware theft, password theft and phishing emails, while the counter examples for availability includes the period of power outages, hardware failures, network interruptions and system upgrades. In following sections, we discuss the security threats of MLBSS from the system aspects of confidentiality, integrity and availability.

#### 3.1 Threat Model for MLBSS

To design a secure software system, a list of system security questions can be utilised, in which one prominent structured approaches is ‘threat modelling’. Threat modelling demands a thorough consideration of ‘who’, ‘what’ and ‘how’, which could structurally represent the available information for security analysis. Considering the MLBSS as a specific class of software systems, the questions will focus on who might want to abuse the system, what capabilities they will have, how they might be able to do it and what the overall risk of the exploitation is. System designers and engineers will be acknowledged with the output about where defence efforts should be focused and what compromises can be anticipated.

Fig. 2 illustrates the five stages of the MLBSS lifecycle, which is extended from the machine learning workflow [2], covering the stages of problem definition, data management, model construction, system deployment and system maintenance.

While traditional cyber attacks may still be effective targeting on MLBSS, we aim to cover the machine learning-based components and processes as the novel attack surfaces, in which attackers can utilise the model and supporting

infrastructure defects affecting both software and hardware components at different development stages. The ultimate goal is to manipulate data, model and system. In comparison to the existing literature reviews, this work considers the security threats from a holistic view to analyse the overall system. It thus covers the security vulnerabilities from data attack, model attack and system attack. Moreover, the correlation among different types of attacks is investigated by providing the engineering practices discussion shown in Fig. 2. It presents a broader and deeper discussion from the the system security point of view, which can provide a clear taxonomy to better serve the system design, testing and auditing goals. Fig. 3 illustrates the current taxonomy for attack types of MLBSS.

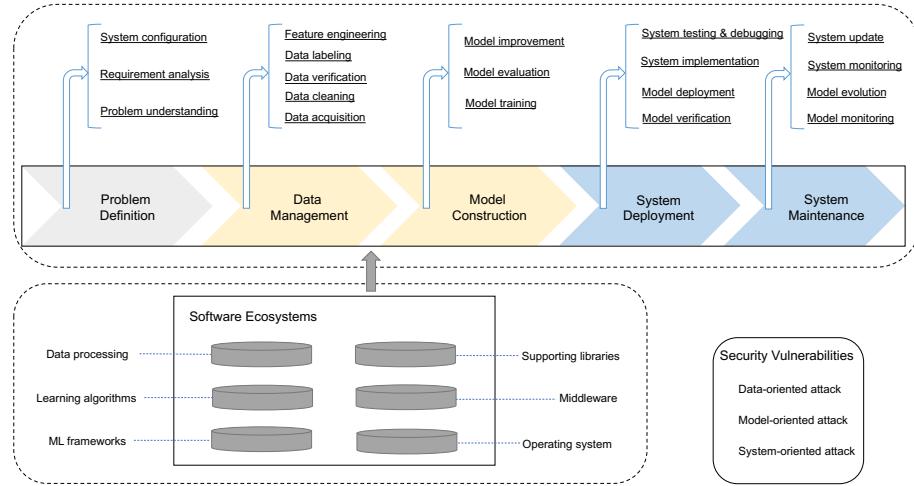


Fig. 2. The five stages of the machine learning-based software system lifecycle

### 3.2 Data-oriented attacks

As per the phases in Fig. 2, data is one of the pivotal assets in MLBSS, as well as the most concerned asset to security threats. Different from traditional software systems, data in MLBSS mostly refer to the training and serving data, whose quality is significant to the system performance. This is also a recently hot topic in modern data-driven systems, such as in [39] the successful deployment and accurate operation of system depend on the continuing conformance of the data to its initial design. Learning and extracting the latent relationships among the large amount of data is the general process of machine learning models. As an accessible vulnerable asset in system, **mislabeled and inferior data** is considered as one passive factor that impact the machine learning model security [55]. On the other hand, attackers can leverage different mechanisms to **access** the training data, **poison** the data, even **inject** the Trojan data.

We measure the data-oriented attacks with respect to the adversary knowledge levels and adversarial goals that may directly impact the training and serving data. In summary, the main training processes include the offline settings (traditional learning on the collected data) and the online settings (such as active learning), which can be achieved by either a centralised way or distributed way. Active learning is designed for achieving higher performance with fewer data samples by querying the labels from an trusted oracle, e.g., an expert annotator [165]. For the latter distributed way, a notion is formalised as federated machine learning, which trains the model with multiple decentralised computational

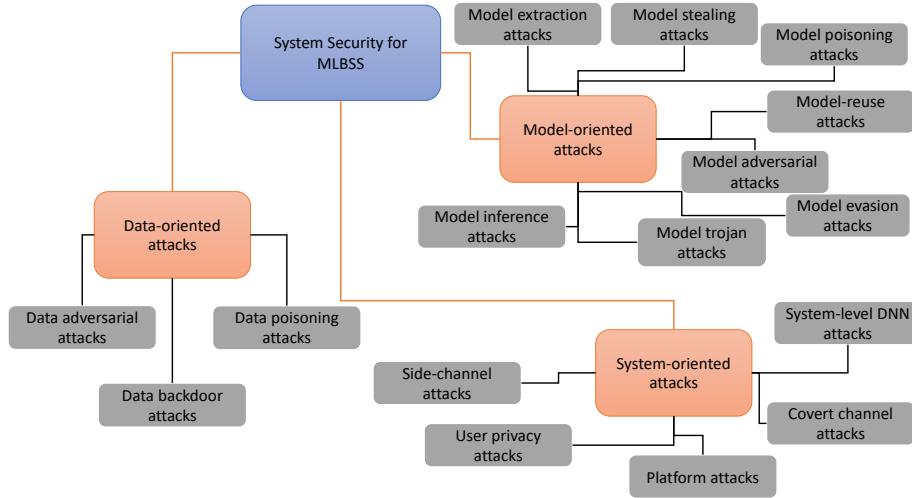


Fig. 3. The taxonomy for system security of machine learning-based software systems

holding local data samples [199]. The settings have a larger liability of exposing the attack surface to the adversary, in which the adversary can manipulate the training data in different stages. In general, the success of machine learning algorithms assumes the integrity of the training data and serving data. Availability of the training data is one main concern for security. The goals of data-oriented attacks can be summarised as to: 1. degrade the machine learning model performance; 2. manipulate the prediction outcome at test time.

#### Review & discussion:

Overall, the data-oriented attacks could be categorised into three major types of attacks, which are poisoning attacks, backdoor attacks and adversarial attacks. While poisoning attacks have been a general term summarising most popular attacks, in this work poisoning attacks represents the triggerless poisoning attacks while the backdoor attacks is the backdoor poisoning attacks. Triggerless poisoning attacks refer to successfully mislead the model and system behaviour without manipulating of the data during model inference stage. For backdoor attacks, the attack will be activated when the input data contains crafted trigger, which could be pixel-level features in an image or designed characters in a sequence. In Table. 2, we have collectively summarised 15 most recently exemplar studies to cover the modes targeting on different development stages.

For both triggerless poisoning attacks and backdoor poisoning attacks, the attacks mostly happen during the data management stage when the attackers could easily obtain the access to the dataset. This assumption is made against the centralised machine learning models [7, 71, 91, 161, 172, 182, 190], whilst one specific scenario is identified for the distributed machine learning models. In [129, 176], the federated learning-based systems are studied that the global model is poisoned by aggregating the edge node model updates learned from malicious participants. The last attacking stage for backdoor poisoning attacks is discovered in [88], in which the model is retrained periodically at system maintenance stage. Generally, attacks have impacts on the data integrity and availability.

Although several earlier works [120, 128, 143, 153] have identified the poisoning attacks in different applications, one earliest work presenting a more realistic assumption by limiting the adversary's capability and knowledge of the system was led by Suciu et al. [172]. Neither the feature nor the algorithm knowledge was taken for granted as the

adversary is limited with four constraints to generate the poison samples in a more inconspicuous way. StingRay attack was proposed as a *model agnostic black-box poisoning attack* against four different machine learning-based software systems for three classification models. In another work from Jagielski et al. [71], linear regression learning models, one of the efficient algorithms in machine learning, were firstly studied against *poisoning availability attacks*. Two attack methodologies, *optimization-based poisoning attacks* (*OptP*) and *statistical-based poisoning attacks* (*StatP*), were developed, while one novel defence algorithm named *TRIM* was evaluated on three public regression datasets. Federated machine learning is one emerging paradigm which benefits from its design of distributed learning of large scale deep learning model and large amount of data. However, federated machine learning also introduces new vulnerabilities while providing user privacy. The AI chat bot *Tay* by Microsoft was reported to be manipulated by malicious users from its online supervised learning model to learn offensive and racist language [159]. In [176], the role of malicious participant contributing to federated machine learning model construction was discussed in the specific label flipping attack, in which the global model was poisoned by the malicious model updates.

In order to understand the threat of poisoning attacks, Schwarzschild et. al [161] has recently developed a benchmarking framework to evaluate a wide range of poison attacks on machine learning models for images. Correspondingly, another work by Wang et al. [182] studied the robustness of stochastic gradient descent to data poisoning attacks in a two-layer neural network. Both works have identified that the machine learning models optimised with stochastic gradient descent (SGD) could be significantly harder to poison.

Backdoor poisoning attack, another type of *poisoning integrity attack* aiming to manipulate the training data to influence the prediction outcome at test time, has emerged as a new form of data poisoning attacks. Sometimes, it is identified as ‘Trojan attacks’. Similar to other data poisoning attacks, attackers will have access to the training data. Adding the stealthy trigger is the main approach, i.e., local patch in the image classification case. One example was studied in Barni et al. [7]. Specifically, a more stealthier way by target data corruption without label poisoning was proposed and evaluated against convolution neural network (CNN) model. In the work by Xiang et al. [190], defence strategy was evaluated given a trained DNN which may use a possibly backdoor-poisoned training data for learning. While data could be distributed in different computing nodes for federated machine learning either vertically or horizontally, Nuding et al. [129] studied the backdoor attacks from two different strategies of *returning model learned from malicious node* and *model replacement from backdoor node* to influence the final model.

Another major type of data-oriented attacks is adversarial attack. The primary goal is to degrade run-time performance when systems are deployed. Thus, how to generate the malicious serving data to yield misleading results while remaining imperceptible is the key. For adversarial attacks, limiting the capability of the adversary to access the information of systems in a black-box setting is a more practical way when attacking system. One early work by Papernot et al. [138] studied to craft adversarial examples by a surrogate training model using synthetic data generation. Attacks were deployed against Amazon Machine Learning and Google’s Cloud Prediction API platforms. Ilyas et al. [68] characterised three more restrictive scenarios. The threat models were defined with query-limited setting, partial-information setting and label-only setting. The novel adversarial attacks were successfully evaluated against a commercial system, which is the Google Cloud Vision (GCV) classifier. While many works focus on image-oriented systems, Boucher et al. [15] recently investigate vulnerability in text-based natural language processing (NLP) systems. For the deployed commercial systems from Microsoft, Google, Facebook and IBM, they have identified that a thorough input sanitization against adversarial attacks is demanded. A most recent adversarial attack example is the physical-world attack investigated by Sato et al. [158]. A *Dirty Road Patch* attack was devised on the real-world automated lane centering (ALC) system,

Table 2. Summary of the surveyed works for the *Data-oriented attacks* for MLBSS.

| Study                          | Attack mode                | Attack stage       | Model   | Affecting CIA                  | Defence                                |
|--------------------------------|----------------------------|--------------------|---|--------------------------------|--|
| Suciuc et al.<br>2018 [172]    | <i>Poisoning attack</i>    | Data management    | CNN, linear SVM and random forest   | Data integrity                 | None                                   |
| Jagielski et al.<br>2018 [71]  |                            | Data management    | OLS, Ridge regression, LASSO and Elastic-net regression   | Data availability              | TRIM algorithm                         |
| Tolpegin et al. 2020 [176]     |                            | Model construction | CNN   | Data & Model integrity         | Identify malicious model updates       |
| Wang et al. 2021 [182]         |                            | Data management    | Neural Network  | Data integrity                 | SGD                                    |
| Schwarzchild et al. 2021 [161] |                            | Data management    | CNN   | Data integrity                 | None                                   |
| Barni et al. 2019 [7]          | <i>Backdoor attack</i>     | Data management    | CNN   | Data integrity                 | None                                   |
| Xiang et al. [190]             |                            | Data management    | DNN   | Data integrity                 | Unsupervised anomaly detection defence |
| Nuding et al. 2020 [129]       |                            | Model construction | CNN   | Model integrity                | None                                   |
| Li et al. 2020 [91]            |                            | Data management    | CNN   | Data integrity                 | Transformation-based defence           |
| Li et al. 2021 [88]            |                            | System maintenance | DNN   | Data integrity                 | None                                   |
| Papernot et al. 2017 [138]     | <i>Adversarial attacks</i> | System deployment  | DNN, logistic regression, SVM, decision tree and nearest neighbors  | Data integrity                 | None                                   |
| Ilyas et al. 2018 [68]         |                            | System deployment  | CNN   | Data integrity                 | None                                   |
| Boucher et al. 2021 [15]       |                            | System deployment  | Fairseq [134], Toxic Content Classifier <sup>1</sup> , Google's Perspective API <sup>2</sup> , RoBERTa [97] | Model integrity & availability | None                                   |
| Sato et al. 2020 [158]         |                            | System deployment  | DNN   | Data integrity                 | None                                   |

which was found to be robust to various real-world factors and general to multiple model design. Meanwhile, it is also stealthy from the driver's view.

### 3.3 Model-oriented attacks

We consider the model-oriented attacks with respect to the adversary knowledge level and adversarial goals that can directly impact the learning models during the training and run-time without accessing the training data. In summary, the adversary will have access to: 1.) modify the training processes, 2.) manipulate the deployed model. To modify the training processes, the adversary has several options for different development stages. For model construction stage, the distribution of the pre-trained malicious models and the modification of the objective function are two main methods. An adversary can also inject security vulnerability during system deployment by adding model patches [31, 83]. The other conventional attacks to manipulate the deployment model will be related to the privacy information leakage, in which the adversary targets on the extraction and inversion of the systems information. The adversary will target data memberships or properties information for model inversion attacks. In terms of model extraction attacks, the primary

goal is to duplicate the deployed machine learning models. The inversion attacks can be launched in a white-box and black-box setting, while the extraction attacks mostly assume a black-box setting.

#### **Review & discussion:**

In summary, the model-oriented attacks are categorised into seven types with different goals, which are neural Trojan attacks, model poisoning attacks, model inference attacks, model adversarial attacks, model reuse attacks, model stealing attacks and model extraction attacks (Fig. 3). In table. 3, we have collectively listed 30 most related studies to demonstrate the attack types for different development stages.

For model poisoning attacks, the primary goal is to undermine the expected performance thus leading to the unusability of trained model. It normally happens in the model construction stage. Both model integrity and availability will be affected. One way to inject the vulnerabilities to the machine learning models is via the distributed pre-trained models. Kurita et al. [83] studied the situation that an attacker has no knowledge of the fine-tuning details but could initialise attacks by accessing the public datasets, which is realistic practice. The model can be exploited even after fine-tuning for the specified task. Another scenario is the distributed machine learning models. When the industry scale MLBSS has now become distributed, a robust distributed machine learning-based system will require Byzantine failure tolerance [13, 109]. An attack will affect the model availability, as the model will end up with denial-of-service attacks, particularly for federated machine learning-based systems [37]. Also, the model poisoning attacks can threaten the learning process directly. In [11], the stealthy model poisoning attack was launched by optimising the malicious objective which is designed for targeted misclassification. While it showed that simply poisoning the data label may not be effective for federated machine learning, the model poisoning attacks appear to be more challenging since two different Byzantine-resilient aggregation mechanisms can not ensure that models will sustain a good performance under attack. Recently, Lorenz et al. studied two direct and indirect poisoning attacks towards deep neural network, which have either modified the training process or poisoned the data [101]. The evaluation of the network certification robustness against training time attacks indicates that better defence methods are required.

For neural Trojan attacks, it also refers to the model poisoning attack but it impacts the model behaviour at run-time when the trigger pattern specified by the attacker is presented. Its wide concerns have been raised due to the available pre-training process could not ensure its integrity. Either the deployed systems are supported by a ML model intellectual property vendor or the malicious patches on deep learning models, its performance on the specific trigger patterns will be largely degraded. While data-oriented attacks have an assumption that attackers have access to modify the training data directly, one type of neural Trojan attacks envisions the attackers benefit from the machine learning model outsourced training, such as the machine-learning-as-a-service provider [96, 99]. Since current neural networks have become larger and bigger, the neural Trojan attacks have become pervasive. Liu et al. [99] demonstrated the security threat of neural Trojan embedded in a neural network intellectual property (IP), which was trained by an IP vendor. Three techniques were proposed to mitigate the identified attacks. The security implications of using third-party primitive models were investigated in [72]. A broad class of model-reuse attacks was demonstrated with four distinct machine learning-based systems from the areas of skin cancer screening, speech recognition, face verification and autonomous driving [72]. Feature extractors were leveraged to trigger the host systems to malfunction. Gu et al. [50] studied the backdoor attack by modelling two different attack scenarios, one was via fully outsourced training task and another is by maliciously pre-trained model from the Caffe Model Zoo and Keras Pre-trained Model Library. The increasingly general practices of outsourcing the training task due to the high demand computational resource and acquiring the pre-trained models from publicly available machine learning model zoo have brought new types of attack surface. It has now become a critical topic to guarantee the model integrity to adopt novel security protocols accordingly.

Chen et al. [24] studied the problem of detecting whether a provided deep neural networks (DNN) model has been trojaned. One defence method name ‘DeepInspect’ was developed to learn the probability distribution of the embedded triggers for the queried models. Another approach ‘TABOR’ developed by Guo et al. [52] targeted different sizes, shapes and locations of triggers pertaining to the trojan and utilised the optimization regularisation technique to better detect trojans. One most recent study by Costales et al. has introduced another type of new live Trojan attack that can directly patch the model parameters at run-time [31]. By minimising the number of overwrites of model parameters in memory, the new live Trojan attack is considered as one of the stealthiest way to bypass the existing intrusion detection systems.

Different from manipulating the MLBSS behaviour at run-time, other two types of model-oriented attacks are model inversion attacks and model extraction attacks. They share same characteristic to proactively extract the privacy information from systems. For model inversion attacks, [5] is one of the earliest works that identified the training dataset leakage. The evaluation on an Internet traffic classifier based on SVM and a speech recognition software based on Hidden Markov models illustrate that, the release of trained machine learning model to the public is not safe. It may result in the information leakage of training dataset. Particularly, the defence method of differential privacy was proved to be ineffective in the scenario. In [167], a shadow training technique is devised to initialise the membership inference attacks against the black-box models training using Amazon ML and Google Prediction API. Shokri et al. have focused on determining whether one acquired data sample was used for the model training. As a mitigation strategy, the regularisation approach is evaluated to be necessary and useful. Salem et al. [155] extended the work by relaxing the assumptions from [167], which are establishing shadow models with same structure as the target model and using a shadow training dataset with same distribution as target training dataset. The result shows that the membership inference attacks can be easily achieved with fewer limitations. These two studies have all applied to machine-learning-as-a-service (MLaaS) providers. One of the work for the membership inference attacks against federated machine learning-based system, or earlier termed as distributed deep learning-based system, was by Hitaj et al. [59]. Generative adversarial networks (GAN) was employed to generate the private information from the trained deep neural network. Though differential privacy (DP) was introduced, it was not investigated if it will be effective for the new type of active inference attack. In a practical setting, Shi et al. [166] investigate the wireless systems, which utilise machine learning models for wireless signal classifier. User authentication in 5G or IoT systems is used as an example. A surrogate model, namely a functionally equivalent classifier, is built as the basis of the membership inference attacks. Another work from [23] studies the inference attacks against the deep generative models based on four different experiment settings, which are accessible discriminator, white-box generator, partial black-box generator and full black-box generator. Although it is not evaluated in a real-world setting, it presents a first taxonomy for membership inference attacks against GAN models. A more realistic threat model is discussed recently in [27], which assumes the only access to hard labels of the trained model. It was expected this type of label-only membership inference attacks could be mounted on any machine learning-based systems. The evaluation of defence strategy suggests that at present, differentially private training and strong L2 regularization will be the most effective defence methods. Another type of inference attacks is the property inference attacks. Ganju et al. [43] proposed to investigate the fully connected neural networks (FCNNs) against property inference attacks. Various data properties were inferred, such as the disproportion of data samples and whether the training data is noisy.

Some works have termed the reverse-engineering work of black-box models as *model extraction attacks*. One early work is by Tramer et al. [178], who have further explored the scenario when attackers only have the query access but no prior knowledge of the models or data. The simple yet efficient attacks were devised to extract targeted ML models including logistic regression, neural networks and decision trees. Furthermore, the online services of BigML

and Amazon Machine Learning were attacked, which indicated enhanced countermeasures will be needed to ensure the ML-as-a-service providers avoid model extraction attacks via exploiting common prediction APIs. Considering the architecture, optimisation process and training data as the model attributes, Oh et al. [132] further explored the model extraction attack for the trained neural network and deep learning models. Juuti et al. [75] have subsequently evaluated different system factors that may relate to the success of DNN model extraction attack, and found that neither hiding hyperparameters of the target model nor reducing DNN outputs from classification probabilities to labels only could contribute to defence.

Similar to model extraction attacks, *model stealing attacks* is another major type of model-oriented attacks targeting on the confidential component, i.e., the model hyperparameters. Wang et al. [181] firstly considered the machine learning models including ridge regression, logistic regression, SVM and neural network as the attacking targets in a white-box setting. The threat model assumes an attacker has access to the training dataset, ML algorithm and optionally to the model parameters. The attack was further evaluated on the machine-learning-as-a-service cloud platform Amazon Machine learning and the hyperparameters value in the objective function were successfully obtained. Keeping the model architectures information and limiting the adversary access to natural seed samples are effective ways to limit the effectiveness. Thus, a generic defence approach ‘Protecting against DNN Model Stealing Attacks’ (PRADA) to detect the model extraction attacks was proposed to detect attacks spanning several queries. The model stealing attacks targeting on the model functionality was also designed with the black-box setting in [133]. The knockoff models give the adversary the advantages to bypass the monetary cost and intellectual efforts. A real-world image recognition API was introduced for further evaluation and the results showed strong performance of the devised knockoff model. In [29], a copycat CNN model attack was presented by utilising random non-labeled data in two steps.

In the work by Papernot et al. [137], two different attacks modes were investigated, which are the model extraction attacks and adversarial attacks. The experiments showed the applicability of DNN and logistic regression (LR) models to learn a substitute model for different classifiers, i.e., deep neural network, logistic regression, support vector machine, decision tree and nearest neighbors. Meanwhile, the online classifiers trained by Amazon and Google were also attacked to misclassification in a black-box setting using a logistic regression substitute model. Huang et al. [64] extended the *adversarial attack* to the pre-trained models from TensorFlow Hub and successfully attacked all 10 representative real-world Android apps with the devised attack. The adversarial examples indicated a practical approach to attack the deep learning models while knowing their pre-trained models for TFLite models. In the work by Hitaj et al. [60], evading the machine learning model watermarking technique is one critical step to avoid the detection and verification by legitimate owners of the stolen machine learning model, which is categorised as the model evasion attacks.

Table 3. Summary of the surveyed works for the *Model-oriented attacks* for MLBSS.

| Study                    | Attack mode              | Attack stage       | Model                        | Affecting CIA      | Defence                     |
|--------------------------|--------------------------|--------------------|------------------------------|--------------------|-----------------------------|
| Blanchard et al. [13]    | <i>Poisoning attacks</i> | Model construction | Distributed machine learning | Model availability | None                        |
| Bhagoji et al. 2019 [11] |                          | Model construction | Federated machine learning   | Model availability | Krum&Coordinate-wise median |
| Fang et al. 2020 [37]    |                          | Model construction | Federated machine learning   | Model availability | ERR&LFR                     |

*Continued on next page*

Table 3 – *Continued from previous page*

| Study                            | Attack mode                     | Attack stage       | Model   | Affecting CIA                  | Defence   |
|----------------------------------|---------------------------------|--------------------|---|--------------------------------|---|
| Kurita et al.<br>2020 [83]       | <i>Model-reuse attacks</i>      | Model construction | BERT  | Data & Model integrity         | SHA hash checksums&Label flip rate                      |
| Lorenz et al.<br>2021 [101]      |                                 | Model construction | DNN   | Model integrity & availability | Fine-pruning  |
| Ji et al.<br>2018 [72]           | <i>Model-reuse attacks</i>      | Model construction | DNN   | Model integrity                | None  |
| Liu et al.<br>2017 [99]          | <i>Trojan attacks</i>           | Model construction | SVM, decision tree                                    | Model integrity                | Input pre-processing, anomaly detection and re-training |
| Chen et al.<br>2019 [24]         |                                 | System deployment  | DNN   | Model integrity                | DeepInspect   |
| Gu et al.<br>2019 [50]           |                                 | Model construction | CNN   | Model integrity                | None  |
| Guo et al.<br>2019 [52]          |                                 | System deployment  | DNN   | Model integrity                | TABOR   |
| Costales et<br>al. 2020 [31]     |                                 | System deployment  | CNN   | Model integrity                | None  |
| Ateniese et<br>al. 2015 [5]      |                                 | System deployment  | SVM, Hidden Markov models                             | Data confidentiality           | None  |
| Hitaj et al.<br>2017 [59]        | <i>Model inference attacks</i>  | Model construction | Distributed deep learning                             | Data confidentiality           | None  |
| Shokri et al.<br>2017 [167]      |                                 | System deployment  | CNN   | Data confidentiality           | Regularisation  |
| Ganju et al.<br>2018 [43]        |                                 | System deployment  | FCNNs   | Data confidentiality           | None  |
| Salem et al.<br>2019 [155]       |                                 | System deployment  | CNN   | Data confidentiality           | Dropout & model stacking                                |
| Shi et al.<br>2020 [166]         |                                 | System deployment  | DNN   | Data confidentiality           | None  |
| Chen et al.<br>2020 [23]         |                                 | Model construction | GAN   | Data confidentiality           | Differential privacy                                    |
| Choquette<br>et al.<br>2021 [27] |                                 | Model construction | DNN   | Data confidentiality           | Differential privacy, strong l2 regularization          |
| Tramer et al.<br>2016 [178]      |                                 | System deployment  | Logistic regression, neural network and decision tree | Model confidentiality          | None  |
| Oh et al.<br>2019 [132]          | <i>Model extraction attacks</i> | System deployment  | CNN   | Model confidentiality          | None  |
| Juuti et al.<br>2019 [75]        |                                 | System deployment  | DNN   | Model confidentiality          | PRADA   |
| Correia et al.<br>2018 [29]      |                                 | System deployment  | CNN   | Model confidentiality          | None  |

*Continued on next page*

Table 3 – *Continued from previous page*

| Study                      | Attack mode  | Attack stage      | Model   | Affecting CIA                         | Defence      |
|----------------------------|--|-------------------|---|---------------------------------------|--------------|
| Krishna et al. 2020 [80]   |  | System deployment | BERT  | Model confidentiality                 | Watermarking |
| Jagielski et al. 2020 [70] |  | System deployment | DNN   | Model confidentiality                 | None         |
| Wang et al. 2018 [181]     | <i>Model stealing attacks</i>                      | System deployment | Ridge regression, logistic regression, SVM and NN | Model confidentiality                 | None         |
| Orekondy et al. 2019 [133] |  | System deployment | CNN   | Model confidentiality                 | None         |
| Papernot et al. 2016 [137] | Adversarial & Model extraction attacks             | System deployment | DNN, LR, SVM and decision trees                   | Model confidentiality; data integrity | None         |
| Huang et al. 2021 [64]     | Adversarial attacks                                | System deployment | DNN   | Model integrity                       | None         |
| Hitaj et al. 2019 [60]     | Evasion attacks (Ensemble attack, detector attack) | System deployment | CNN   | Model confidentiality                 | None         |

### 3.4 System-oriented attacks

Till now, the analysis of security threats against MLBSS centers on the data, algorithmic and model aspects. As we have discussed in previous sections of ‘data-oriented attacks’, ‘model-oriented attacks’, in this section, the focus is on the vulnerabilities from the constructional and functional components carried with the system. Since MLBSS will harness more dedicated and dynamics components in terms of hardware and software to fulfill the dedicated functionalities, these components are tightly connected to achieve the legitimate operations in a valid and sustainable way, i.e., concerning the increasing computation and storage requirements, and the run-time environment. The traditional security threats have become much more complicated in the MLBSS, in which new attack surface can be from the middleware, the globalised supply chain, the runtime engine and so on. However, the system-oriented security threats often emerge through the interplay of multiple systems while overseen until they are actively exploited. While the attackers may leverage various attacking surfaces, including hardware, in this section, we will discuss recently identified system-oriented attacks.

#### Review & discussion:

In this work, we firstly measure the hardware-related system-oriented attacks with respect to the goals and capabilities that alter the model and data information via manipulating the hardware-related software and channels, i.e., the middleware. One instrumental asset that largely contributes to the success of MLBSS is the advanced computational infrastructure, i.e., the specialised hardware accelerators. Particularly, recent advancement of machine learning techniques has been well positioned with a large demand of computing power to handle the accumulated big dataset and complicated calculations for deep neural networks. It is now a hot research topic aiming at better leveraging not only the software, but also the hardware, in terms of the design logic to fulfill the proposed functionalities. For example, specialised hardware accelerators are one type of the hardware that upholds the hardware requirements for such resource-demanding applications. Accordingly, system-on-Chip (SOC) design has become a hot topic to ensure

Table 4. Summary of the surveyed works for the *System-related attacks* for MLBSS.

| Study                       | Attack mode                                    | Attack stage       | Model                            | Affecting CIA                     | Defence  |
|-----------------------------|--|--------------------|----------------------------------|-----------------------------------|--|
| Song et al.<br>2017 [169]   | User privacy stealing                          | System deployment  | CNN, Residual networks, SVM & LR | Data confidentiality              | Data obfuscation   |
| Liu et al.<br>2019 [94]     | System-level DNN attacks                       | Model construction | DNN                              | Model integrity & confidentiality | None   |
| Kim et al.<br>2020 [77]     | Platform attacks                               | System deployment  | CNN                              | Model availability                | Data-space randomization, hardware access controls         |
| Costa et al.<br>2021 [30]   | Covert channels attacks                        | Model construction | Federated machine learning       | Model integrity                   | None   |
| Xu et al.<br>2021 [195]     | Trojan, side-channel & fault-injection attacks | Model construction | Neural network                   | Model integrity&availability      | Yes  |
| Mittal et al.<br>2021 [117] | Trojan, side-channel & fault-injection attacks | Model construction | DNN                              | Model integrity & availability    | Yes  |
| Zhu et al.<br>2021 [213]    | Side-channel attacks                           | System deployment  | DNN                              | Model confidentiality             | Encryption, obfuscation, adding noise&offload tasks to CPU |

the model performance and improve the energy efficiency, which has exposed further stealthy attack surfaces, i.e., implanting the hardware trojan in the Graphical Processing Units (GPUs) and Tensor Processing Units (TPUs). Such an attacker can leverage the hardware attack vectors to achieve the goals of model corruption, backdoor insertion, model extraction, spoofing, information extraction and sybil attack [195]. Generally, the traditional and popular process for system designers to solve the hardware-software co-design problem is to integrate specialised hardware accelerators from external hardware designers. This results in a significant concern of trust, since outsourced accelerators could have malicious modifications, also known as hardware Trojan. Several works have extensively discussed this matter covering the traditional ASIC and FPGA architectures [28, 63, 98, 100, 179, 196, 197, 201]. Given the continuous development and progress of hardware-software co-design for MLBSS, the hardware-oriented approaches attempting to steal or corrupt the hardware thus impacting the MLBSS have seen its severity.

Furthermore, we specifically focus on the hardware-related software attack vectors. One extraction attack on the unencrypted PCIe traffic is summarised in Table. 4 [213]. The unencrypted PCIe traffic is a novel side-channel attack surface [195, 213]. Different from general model extraction methods, the hardware model stealing attack assumes that the adversary has physical access to GPU devices, particularly to the PCIe bus. A novel Hermes Attack was proposed to fully reconstruct the whole DNN models in a black-box setting. The last group of hardware-related attacks is discussed in [117, 195], which is side-channel attacks. The goals of side-channel attacks can be various, including model extraction from PCIe [213] and GPU [121, 184].

In comparison to the algorithmic attacks for machine learning-based software systems, another observation is that the system-oriented attacks can also be easily integrated with traditional attack vectors and approaches like malware injection and user privacy stealing. Since machine learning has become a commodity, Song et al. [169] studied the scenario that machine learning code was provided by a malicious service provider. It was identified that, no matter the adversary is on white- or black-box mode, the provided machine learning code can be the attacking surface that “memorise” the sensitive information from the user in the final model. The model was still proved to be accurate and

meet the functional requirements. However, it leaves the exploitable point for the adversary to obtain the privacy. It would be relatively difficult for the success of manual inspection of the source code. One suggested manner from [169] is to turn the least significant bits attacks against itself. In [94], an overview of the system-level attacks is discussed, particularly focusing on malware injections. An assumption was made that the adversary may have the exclusive access to the models, as well as the supporting hardware and software components in the corresponding execution engine. Following, Kim et al. [77] further elaborated the attacks towards AI platforms. A memory safety vulnerability in a third party library was investigated which has caused the targeted misclassification for the machine learning system. The open-source toolkit, namely Microsoft CNTK, was leveraged and the exploit relied on the identified vulnerability of CVE-2018-5268. Recently, Costa et al. [30] utilised the federated machine learning system to introduce a new covert channels attack. The stealth communications between learning agents were established by violating the isolated training protocol, which could lead to the user privacy stealing and model manipulation.

### 3.5 Summary

In this section, we have collectively surveyed the security threats for machine learning-based software systems with a holistic system view, which is illustrated as well in Fig. 3 covering data- (15 papers), model- (30 papers), and system-oriented (7 papers) attacks. It is noted that, current the attacks and defences for MLBSS are more and more similar as the cat and mouse games in traditional system security area. While the collection of the papers can be more due to the popularity of this topic, we anticipate that the collective discussion in this section covers the major distinguishable artefacts of MLBSS. The attacks modes are subsequently categorised in terms of the direct or most related artefact.

Following the studies from [114, 145], Table. 5 is a summarised two-level hierarchy taxonomy of vulnerabilities causes for general internet software systems [145]. While the classical taxonomy is not a static item, it serves as a general reference to classify the software security threats in terms of software development practice, which are secure input and output handling, race condition, faulty use of API, memory safety, improper use case handling, improper exception handling, resource leaks, and pre-processing input strings [145, 185]. Thus, the risks of MLBSS depicted in Table. 6 [114] are exclusively covered and extended in the discussion of the literature in this section with a taxonomy summary of Fig. 3. Although [114] presented the analysis mostly from a general term, we have substantially extended the scope and distill the available details from relevant literature to justify the emerging and novel attack modes.

In the following sections, we will continue the investigation with security threats with the traditional software engineering practices for MLBSS as many challenges have been proposed. With the discussion from the perspective of affecting developing stage, it is anticipated to understand the state-of-the-practice of secure development for MLBSS. A major part of the discussion will subsequently focus on the secure development practice, introducing the main strategies for MLBSS development lifecycle.

## 4 SECURE DEVELOPMENT PRACTICES FOR MLBSS

To understand the security threats for MLBSS, a thorough review via the traditional confidentiality, integrity and availability (CIA) model has been implemented with respect to the attacks types towards the dominant artefacts in MLBSS in Section. 3 (see Table. 2, 3, and 4). Meanwhile, the attacking stages are discussed considering the MLBSS system development lifecycle from problem definition, data management, model construction, system deployment to system maintenance. It is observed that current attacks have a major focus on data management and model construction

Table 5. A Two-level hierarchy taxonomy of vulnerabilities causes for traditional internet software systems [145]

| Phases               | Vulnerabilities Causes                      |
|----------------------|---|
| Analysis phase       | No risk analysis/ No security policy        |
|                      | Biased risk analysis                        |
|                      | Unanticipated risks                         |
| Design phase         | Relying on non-secure abstractions          |
|                      | Security/Convenience tradeoff               |
|                      | No logging                                  |
|                      | Design does not capture all risks           |
| Implementation phase | Insufficiently defensive input checking     |
|                      | Non-atomic check and use                    |
|                      | Access validation errors                    |
|                      | Incorrect crypto primitive implementation   |
| Deployment phase     | Insecure handling of exceptional conditions |
|                      | Bugs in security logic                      |
|                      | Reuse in more hostile environments          |
| Maintenance phase    | Complex or unnecessary configuration        |
|                      | Insecure defaults                           |
| Maintenance phase    | Feature interaction                         |
|                      | Insecure fallback                           |

Table 6. List of Top Ten Risks of Machine learning-based Systems [114]

|                            |                          |
|----------------------------|--------------------------|
| Adversarial examples       | Data poisoning           |
| Online system manipulation | Transfer learning attack |
| Data confidentiality       | Data trustworthiness     |
| Reproducibility            | Overfitting              |
| Encoding integrity         | Output integrity         |

stages. It has now aimed for a more stealthier approach and a more critical outcome with either additional access to the side channels or advanced algorithms to learning outcomes. Human factors remain remain the fundamental element of system security [4].

Thus, in this work, we have further investigated how current software engineering practices can support the system security of MLBSS by mitigating the aforementioned various attack types and outcomes. Since the integration of security into the system lifecycle can serve an efficient insertion of security processes into the existing system lifecycle [1, 32, 148], we briefly describe the current software engineering practices for MLBSS in general. A main discussion will follow the MLBSS lifecycle, particularly from the perspective of the security practices.

#### 4.1 Software engineering practices for MLBSS

Whilst the machine learning components have extensively ongoing development and become an integral component in larger software systems, the wide deployment of MLBSS has led the research trends to adopt various software engineering practices. Extending the development process with the essential components from [162], in Fig. 4, the machine learning model development only represents a tiny fraction of whole system. It is undoubtedly that the intertwining process of machine learning and software engineering has opened many challenges. A recently reported industrial case SAP [149] reports the challenges in software engineering.

For traditional software development, the Guide to the Software Engineering Body of Knowledge (SWEBOK) [17] provides a comprehensive umbrella to cover the software engineering topics, including *software requirement*, *software design*, *software construction*, *software maintenance*, *software engineering models and methods*, *software engineering*

*process, software quality, software testing, software engineering management, software configuration management and software engineering professional practice.* To understand the difference and challenges, SWEBOK is introduced as the taxonomy for SE practices classification in [2, 82, 110, 164, 180]. A conclusion is collated that, the challenges presented in the development of MLBSS are specific and can not be well managed within the SWEBOK topics, especially in terms of the technical and data-related challenges. This, in turn, calls for adjusting the practices in the context of system domain and technical issues.

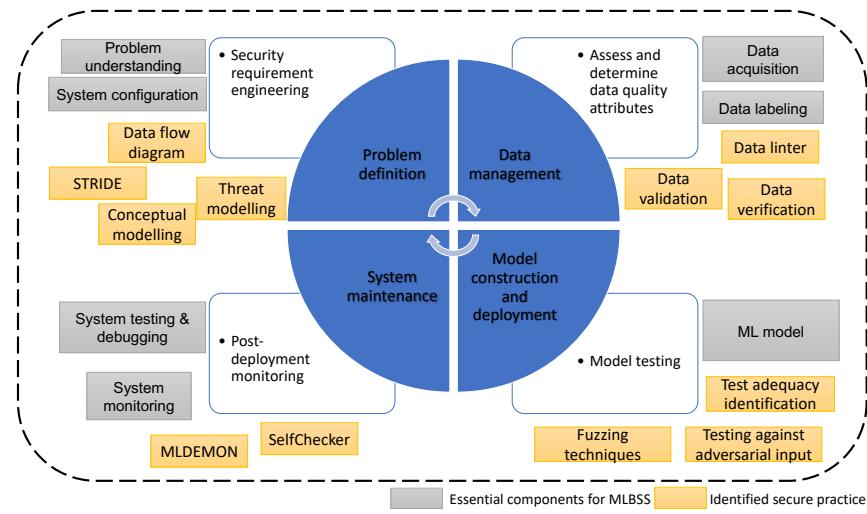


Fig. 4. The essential components [162] and identified secure practices for MLBSS

Currently, the research and development of the state-of-the-practice for MLBSS are mostly focused on two topics, which are technical debt and quality assurance. Generally, quality assurance defines '*a planned and systematic pattern of all actions necessary to provide adequate confidence that the item or project conforms to established technical requirements*' [20]. For technical debt, at its broadest, it was defined as '*any side of the current system that is considered sub-optimal from a technical perspective*' [81, 177]. While the discussion of technical debt is related to the compromised decisions for the poor software development [93], the quality assurance dedicates to system quality offline and in production attached with several distinct aspects, i.e., model quality, safety and fairness analysis [76].

Technical debt, studied in [177] for classical software systems, has recently been reviewed in MLBSS in order to identify the specific patterns [14, 93, 162, 175]. Additional to typical technical debt at code level, the machine learning-based systems represent extra trade-offs to be overcome for practical practices for the long term development, deployment and maintenance at the system level. The data dependency debt was extensively discussed as a key contributor for MLBSS since it is one of the integral components in MLBSS and is generally more difficult to detect as a result of its unstable and underutilised characteristics [162]. Other technical debts, i.e., abstraction debt, configuration debt, data testing debt, reproducibility debt, process management debt and cultural debt, may also be accrued and demands ongoing collaborative efforts. Following in [14], 21 primary studies were included to identify the actual technical debts that have been investigated in recent works, in which four novel emerging debts of data, model, configuration and ethics were elaborated. Taking further actions towards measuring the technical debt and targeting to pay it off, a recent

work by Liu et al. has investigated the deep learning frameworks which is an essential component for both researchers and industry developers [93]. The analysis of seven most popular open-source deep learning frameworks illustrates a demanding understanding and countermeasures for the design debt, defect debt, documentation debt, test debt, test debt, requirement debt, compatibility debt and algorithm debt. In [175], refactoring was empirically studied against MLBSS, particularly to answer the machine learning-specific refactoring. It is found that duplicate code elimination in configuration and model code is the top refactoring for code duplication debt in MLBSS, while more refactoring will be required to technical debt in MLBSS.

Another main topic is about quality assurance. Quality assurance has been well established for traditional software development in terms of proper software quality in conformance with standards and procedures [20]. Ma et al. [104] is the first work that attempts to answer the question of common challenges relevant to quality assurance of deep learning. Particularly, a collection of existing literature was evaluated indicating that there still lacks a thorough understanding of the occurrence of deep learning bugs. Thus, a Secure Deep Learning Engineering Life Cycle (SDLE) was introduced to support the future quality assurance of development. Recently, Hamada et al. [54] further investigated the quality assurance for machine learning-based systems by reporting the initial version of guidelines, which was an extended outcome of QA4AI Consortium held in Japan consisting 39 researchers and practitioners and three different organisations. The common core concepts of the quality assurance for MLBSS were summarised in terms of the axes of quality evaluation. Considering two different software development styles, i.e., deductive style and inductive style, five aspects of quality evaluation for MLBSS, including data integrity, model robustness, system quality, process agility and customer expectation, were extracted.

Including the discussed technical debt and quality assurance for MLBSS, the development of MLBSS remains challenging to define a comprehensive and consolidated engineering protocol, which will ensure a robust and secure development, deployment and maintenance process. Some works have attempted to explore the distinct components from system level. Humbatova et al. looked into the general artefacts, i.e., *model*, *tensor&input*, *training*, *GPU usage* and *API usage*, which may be failed and exploited in deep learning systems from the developer's points of view [65]. Lewis et al. alternatively took the mismatch system elements that may occur during the end-to-end development by different stakeholders into the consideration, in which the trained model, raw data, training data and the environments were identified [87]. However, more efforts are still needed to support the robust and secure engineering of MLBSS. In following subsection, we will reviewed the current security practices for MLBSS.

#### 4.2 Current secure development practices for MLBSS

The goal of system security for MLBSS is to ensure the software systems function in a correct way continuously in the deployed environment [112]. It engages the software engineering practices by distilling the knowledge of security threat and risk analysis in the software development lifecycle [86], such as early on for the security requirements engineering [53, 116]. The practices of security engineering would need to be applied to various software artefacts to understand their adoption and affects accordingly within the system context.

One recent work by McGraw et al. [113] demonstrates that the security risks for MLBSS exhibit a much more complicated scenario for all practitioners. A thorough understanding of the artefacts, along with their inherent relationship between each other and the external interactions with the process, is the main driving force towards securing machine learning-based software systems. McGraw et al. has subsequently focused on the architectural risk analysis for machine learning security, which firstly initiated the work of [115] and further elaborated the identified

top ten risks in [114]. While the analysis output can serve as a risk analysis framework for MLBSS, it lacks an in-depth consideration to design, implement and field the machine learning-based systems.

Noting that a brief discussion of security risks analysis for data, model (development and runtime), input (runtime) from [113], we have further categorised and curated the identified attacks targeting on data, model, and systems through Section 3. This serves a refined representation of these specific attacks from the artefacts perspective, which could be better leveraged for MLBSS secure development practices from the software lifecycle. As illustrated in Fig. 4, current secure development practices for MLBSS also share same conceptual maps to some extent while there remains much more work unexplored to bridge the novel attack modes and challenges. Currently, significant researches targeting secure development practices for MLBSS are under development. Accordingly, we review the work focusing on each development phase, including *security requirement engineering* for problem definition, *data verification and sanitization* for data management, *machine learning model and dependency testing* for model construction and deployment, and *post-deployment monitoring* for system maintenance. We will also include the discussion and vision of these work concerning the mitigation of categorised attacks from the artefacts perspective.

**4.2.1 Problem definition.** The different application trends in the software engineering industry demand different understanding and practice, which propels the development process to be in dynamically changes regarding the daily activities and the practitioners' roles evolution. It is critical to properly define the problem in the context of application. In the stage of *problem definition*, designers and related stakeholders are gathered together to determine the features concerning the functional and non-functional requirements, particular when the systems require machine learning driven modules as the core. This brings the *security requirement engineering* a critical software development process to general information systems [116].

Whilst taking security requirement engineering early on in the modern software development process, the inherent uncertainty of machine learning driven modules is one of the main constraints that impose critical challenges and substantial changes to the existing secure development practice [2]. Recently, several studies have aimed to address the requirement engineering for machine learning, particularly from the security requirement engineering aspect [44, 58, 61, 124, 187, 200].

One early study [61] initiated the discussion of quality-related non-functional requirement for machine learning-based systems, for which significant challenges and gaps were identified for the qualities of machine learning-based systems. It was the first work related to the qualities for MLBSS such as security, privacy and transparency and so on. Wilhjelm et al. [187] firstly extended the security concerns for MLBSS with a holistic paradigm to include the specific and traditional threats. Particularly, Wilhjelm et al. has applied the traditional threat modelling method together with the identified attack libraries to understand the security of MLBSS at the requirements phase [187]. Data flow diagrams and STRIDE methods were utilised as the main methods for security requirements elicitation in practices. STRIDE is one threats classification methodology based on the Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege threat categories. Conceptual modelling is introduced in [123, 125] to provide multi modelling views for systematic analytical requirements elicitation, which are the business view, analytics design view and data preparation view. It is subsequently employed in the health sector [124], though the details concerning the quality attributes or non-functional requirements for users are not thoroughly discussed in terms of the gained advantages via the goal-oriented requirement for machine learning (GR4ML) framework.

Another work from [44] employed a different process defined in [3] to the machine learnt models (MLMs) in self-driving cars. The safety requirements for MLMs were analysed to drive the assurance activities for data management

and model construction phases. It is noted that a rigorous and accurate understanding and definition of the problem could well serve as a mitigation method thwart the success of *data-* and *model-oriented* attacks (see section. 3.2, 3.3). It is demonstrated with a specific task of pedestrian detection at crossings by considering the vehicle-level safety requirements. The MLMs safety requirements for data management phase were subsequently elicited in line with the assurance desiderata, which are relevant, complete, accurate and balanced. However, the details of model construction phase were missing. While there is no other existing work to assure MLMs as well as the released safety standards such as ISO26262, Gauerhof et al. [44] has demonstrated the importance and applicability of designing MLMs safety requirements for the safety assurance activities in data management and model construction phases. Yang et al. considered the specific safety-critical system for machine learning deployment, i.e., the aerospace domain [200]. The restricted natural language requirements modelling process was evaluated with the implementation of an industrial case Autonomous Guidance, Navigation and Control (AGNC). A most recent research from [58] has illustrated that to guarantee the machine learning-based systems behaviour with a well-defined system attributes such as safety, major problem areas including contextual definition and requirements, data attributes and requirements, performance definition and monitoring, and human factors, will need substantial efforts from the requirement and systems engineering research.

**4.2.2 Data management.** Data persistence is important to the entire software systems. For traditional systems, the privacy of the digital information such as user data and system configuration in modern systems is critical and demands thorough and effective security strategies for the entire lifecycle [198]. While it is a general requirement that the data security covers the hardware and general software logic aspects [67], the interaction with machine learning models requires exclusive consideration of data and its probability distribution representation [58]. It is due to the uncertainty of the learnt machine learning models, which brings the data quality [8, 56] as one important aspect during data management stage to ensure the development of MLBSS. Regarding data as the object to be utilised in both design time and runtime, how to assess and determine the data quality attributes in the context of machine learning-based systems carry the significant value for the data-oriented attacks towards a robust and reliable model and system [156, 170, 188].

Overall, there have been some state-of-the-art practices targeting on the data management tasks against data-oriented attacks, such as the enrichment of dedicated data metrics and the study of data and model correlation for the data poisoning attacks [41, 130, 211], a data schematic for adversarial attacks [103, 146, 207] and data practices of lint and strictly controlled labelling process for backdoor attacks [66, 147]. An initial attempt to support the data management task for machine learning-based system in a comprehensive manner is from [146]. A high-level schematic compromising training/serving data and model for a production machine learning pipeline was discussed, which further elaborated four key challenges for data management task, which are understanding, validation, cleaning and enrichment. Data validation is considered as one prominent process to guarantee data quality for MLBSS [103]. While it has potential thwart the data poisoning and adversarial attacks, it remains unknown for the data backdoor attacks. Hynes et al. [66] explored the common data practices for developing machine learning models by introducing the concept of lint for data preparation and feature engineering. Data linter was designed and implemented as a tool to support the analysis of training data for miscoding, outliers and packaging errors, thus the targeting model quality could be ensured and improved to minimise the chances of backdoor attacks. Prendki et al. [147] investigated the data labelling process to be performed in a strictly controlled process by discussing three common practices for reducing the data annotation costs in the lifecycle. While the data quality can not be ensured, i.e., excluding faulty data from different sources, Nurminen et al. [130] studies the correlation between the complex data and different machine learning models to support the

development of robust and reliable MLBSS. Another way to mitigate the low data quality issue is a conceptual risk-based data validation approach [41].

In spite of ensuring the data quality is critical, there are limited research works targeting understanding the violation of CIA model in data management stage. Other contemporary methods to improve the robustness and reliability of machine learning-based systems include introducing extra metrics for the input data [207] and devising a novel verification scheme in the production environment [211]. While the work by Zhao et al. [211] developed a novel data verification scheme for distributed machine learning to ensure the integrity of training data, Zhang et al. [207] considered various uncertainty metrics for the data, particularly for the adversarial examples and benign examples. The idea is to characterise the uncertainty patterns of data for testing. One inspiring finding is that the patterns of the data generated by existing methods follow common patterns. In light of this, a new input data generation method was devised to generate uncommon adversarial examples, which experiments indicated an increasing successful rate against existing defence techniques.

**4.2.3 Model construction and System deployment.** In comparison with traditional software systems, the MLBSS brings the learning advantages as well as the inherent uncertainty, thereby widening the potential attack surface. In Sec. 3, many attackers have demonstrated an advanced leverage of the statistical uncertainty of the trained models from various points, i.e., feature vectors and model training algorithms. Also, the system deployment have introduced novel frameworks and libraries from different sources. From one aspect, the increasing attack surfaces for MLBSS impose specific attacks for existing detection and defence techniques. From another aspect, this brings the challenges to traditional software testing strategy, which is designed as one primary instrument to evaluate system from the safety assurance perspective.

To fulfill the traditional software testing, general practices have to identify testing principles such as test adequacy in compliance with established standards, such as ISO26262 [69] for road vehicles domain. Adopting the testing practice to secure the machine learning-based systems development, particular for deep learning, has been an emerging and challenging topic for model-oriented attacks [18] (see section 3.3).

One studied criteria is related to the testing adequacy. DeepXplore is one of the early works that attempts to automatically generate test inputs for deep learning models to expose erroneous behaviors in an efficient way [141]. Following, different works based on traditional testing adequacy coverage methods are proposed and extensively evaluated, such as the mutation testing [107], combinatorial testing technique [105], multi-granularity testing criteria [106]. In [78], a novel test adequacy criterion for testing called Surprise Adequacy for Deep Learning Systems (SADL) is proposed. To capture the importance of deep learning-based system in a layer-wise functional understanding, [45] has recently proposed an importance-driven test adequacy criterion. Furthermore, it was evaluated against the adversarial generation techniques for the model adversarial attacks. The other criteria could be obtained with the fault coverage for MLBSS. It could be achieved by applying dual modular redundancy to check the fault coverage in the entire DNN model [89, 90]. In [90], the framework of D2NN is proposed for checking the result inconsistency between the original and duplicated models. In another work of [89], a dynamic verification method named DeepDyve is experimented with sufficient fault coverage yet little computational overhead for DNN-based classification systems. Both works have demonstrated the verification of fault tolerant capability in the DNN models could secure the model from external attacks.

In addition to the general test principles coverage, one specific challenge for MLBSS is the robustness for adversarial input. Thus, another group of practices targets on testing MLBSS against adversarial attacks (see section 3.3). Wicker et al. [186] initiated the adversarial examples crafting as a two-player turn-based stochastic game to test the deep neural

network. Subsequently, Ma et al. [108] has identified two common exploitation channels of the provenance channel and the activation value distribution channel in details. A novel invariant based detection technique against the adversarial samples was then proposed. The traditional coverage-guided fuzzing technique was combined with property-based testing to extend the testing coverage on adversarial samples for neural network [131], while in [192] DeepHunter based on same fuzzing technique but different mutation strategies was evaluated against general-purpose DNNs.

The third topic related to the MLBSS is the deep learning libraries, which has recently been extensively studied leading to the model-reuse attacks (see section 3.3) and system-oriented attacks (see section 3.4). Since the libraries are the basic component during the model construction as well as the system deployment, it will cause more critical bugs and be easier to be exploited in comparison to the model and data, such as the TensorFlow [209]. Given the large amounts of different machine learning frameworks and platforms, particularly for deep learning, Guo et al. [51] studied the performance metrics for different frameworks. For the development process, this includes the accuracy, adversarial robustness of trained models based on four representative frameworks, which are TensorFlow 1.12.0, PyTorch-0.4.1, CNTK-2.6 and MXNet-1.4.0. Additionally, Guo et al. also studied performance after model migration and quantization for the system deployment process, such as the migrated model for mobile and web platform. With the observed performance decline for the two different processes, the concern and attention for deep learning software community are raised to address alleviate the deviation amid frameworks and platforms for a universal deep learning solutions. Pham et al. has implemented CRADLE, a cross-backend validation to detect and localise bugs in deep learning libraries [144]. The effectiveness of CRADLE in detecting bugs and unique inconsistencies for TensorFlow, CNTK and theano has called for extra attention for testing deep learning implementations. Similarly, oracle approximations as a common practice was adopted to represent numerical errors in deep learning libraries and was studied in four popular libraries, which are Tensorflow, theano, PyTorch and Keras [127]. LEMON, a deep learning library testing via guided mutation based on novel heuristic strategy is recently proposed by Wang et al. [183] to library testing. It brought the attention to the 24 novel bugs which were identified in 20 published versions of 4 representative deep learning libraries such as TensorFlow, Theano, CNTK, MXNet.

**4.2.4 System maintenance.** For system maintenance, model and system monitoring is critical to the consistent assurance of MLBSS for security, reliability and efficiency. The output must be intensively monitored as a deviated output from a machine learning-based software system, which could not be fully pledged to avoid uncertainty, will cause catastrophic consequences. While the security engineering practices for MLBSS is still in its infancy, several works have attempted to monitor the model update for post-deployment.

In [47], Ginart et al. investigated a much complex scenario for post-deployment monitoring, which lacks sufficient sources for labeled data to estimate the model's on-time performance. Alternatively, the MLDEMON, for ML DEployment MONitoring, was validated on eight realistic data streams to improve the system deployment in terms of reliability and efficiency. Xiao et al. [191] focuses on the DNNs monitoring. In this work, SelfChecker is devised to monitor the outputs and triggers an alarm when inconsistency is detected in the internal layers.

## 5 CHALLENGES AND FUTURE DIRECTIONS FOR SECURITY ENGINEERING OF MLBSS

While current security engineering practices in MLBSS has leveraged a large amount of efforts and resources, it is evident that a wide range of challenges of security engineering remain unexploited. Thus, in this section, we discuss the challenges with the security engineering for MLBSS from different perspectives aiming at the data-related, model-related and system-level framework elements. Accordingly, the future directions are discussed to address the challenges.

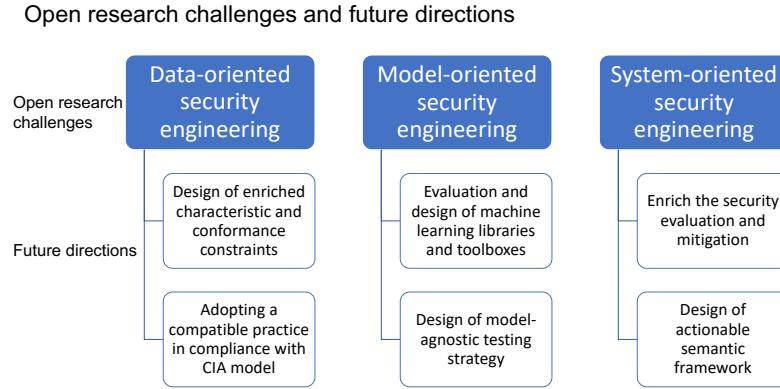


Fig. 5. List of challenges and future directions for security engineering of MLBSS.

### 5.1 Data-oriented security engineering for MLBSS

This section brings attentions to the security engineering for MLBSS in terms of the data-oriented attacks, particularly focusing on the data management stage (section 4.2.2). Firstly, it is recognised that current security practices has not fully considered the data confidentiality and integrity since the lack of an insightful understanding towards the security aspect of data management. Secondly, there is little work for the relaxation of data-oriented attacks in data management stage, which could provide a fundamental protocol (section 3.2). To relax these two challenges, design of enriched characteristic and conformance constraints, and adopting a compatible practice in compliance with CIA model for data management would be potential future directions.

**5.1.1 Design of enriched characteristic and conformance constraints for data.** Data remains the pivotal component for machine learning-based systems, such that the malfunction behaviour of systems can largely reside in the data. When splitting data for the training and evaluation of system deviates from each other, the output will become unreliable and unsafe for safety-critical systems. Thus, the introduction of conformance characteristic for data management stage can greatly alleviate this concern. Furthermore, the conformance should also be applied to the unseen data, which will be used for post-deployment inference against poisoning and backdoor attack.

It is inspiring to adopting a compatible data auditing practice to ensure the data integrity from the work of Zhang et al. [207]. In this work, it indicates that the benign and adversarial data generated by same existing methods could be statistically categorised based on common uncertainty patterns. Thus, the defence capability could be further achieved by devising deviated adversarial data that do not follow the patterns for training, against the adversarial attack. Moreover, in [38, 39], the data profiling is investigated for data-driven systems to discover the conformance constraints for a dataset, with one recent framework proposed in [42] called DataExposer. Hence, future work could be to study how to accurately define the profile, enrich the characteristics and conformance constraints for data understanding, validation, integration and cleaning.

**5.1.2 Adopting a compatible practice in compliance with CIA model for data management.** In section 4.2.2, we can see that a general focus of the current practice is on the adoption of data quality practice. The aim is to ensure the accurate

and robust machine learning-based functionalities during the inference time. While the data management stage involves the training and testing data for subsequent model construction and system deployment, the data should be obtained in a rigorous and standardised manner, indicating that CIA triad can be a better option for data management stage.

Furthermore, the circumstance will become worse when the data management task is largely outsourced. Zhao et al. [211] has explored the provable data possession verification scheme which is generally effective in cloud computing. The effectiveness of the distributed machine learning oriented data integrity verification scheme demonstrates the importance of adopting a compatible practice in the context. Thus, we call for future work to investigate a broader and more effective verification scheme, i.e., blockchain [212], to ensure that CIA model is tightly withhold for the data.

## 5.2 Model-oriented security engineering for MLBSS

For model-oriented security, current understanding and practices are not sufficient to secure the machine learning-based system against the identified attack types. Firstly, the machine learning libraries and toolboxes testing is a relevant topic to test the models yet little attention has been paid to. Secondly, current security practices targeting model testing could not generalise well for different types of machine learning models. Herein, we highlight following two future directions to tackle the challenges – the evaluation and design of machine learning libraries and toolboxes; the design of model-agnostic system testing strategy.

**5.2.1 Evaluation and design of the machine learning libraries and toolboxes:** Besides the training and testing data, there are several complex integral components whilst building MLBSS (see Fig. 2 and Fig. 4). Currently, testing the robustness of the trained models has attracted most attentions to guarantee the quality and security of machine learning models. The most recent work [183] (see section. 4.2.3) has targeted on another venue, which is to test the deep learning libraries including TensorFlow, Theano, CNTK and MXNet. A rather much more complicated and critical machine learning development community in comparison to traditional software development has been presented in this work.

In the meantime, the various machine learning libraries and toolboxes could result in unknown security threats and uncertain performance impact for the machine learning model construction and system deployment stages. It is clear that testing the libraries and toolboxes would not be sufficient. Thus, a comprehensive evaluation for the machine learning libraries and toolboxes could be a more detailed option to alleviate the concern. Interestingly, a recent work from [79] has initiated the discussion of a summary of best practices and a collection of machine learning toolboxes design patterns. It also highlights that it is promising to further extend the design of machine learning libraries and toolboxes following a certain, secure and practical design principles and patterns.

**5.2.2 Design of model-agnostic testing strategy for MLBSS:** Although model testing has been investigated to achieve the testing adequacy with standards for MLBSS, it has been a fundamental challenge to test the learnt model and system against all potential adversarial input [34, 151, 193] (see section. 4.2.3).

Generally, the testing strategy is devised to secure models against adversarial inputs. Recent work from Fan et al. [36] has extended the quality testing for recurrent neural network to backdoor attacks. However, it could be challenging to adopt the available testing strategy for GAN model, such as against membership inference attacks [23]. Thus, a model-agnostic framework and benchmark of testing techniques for the evaluation of diverse attack types against different machine learning models will be much more valuable to collectively support model-oriented security engineering, as a future work.

### 5.3 System-oriented security engineering for MLBSS

The machine learning-based software system is essentially a software product following the identified pipelines (Fig. 2). However, little work has been identified to provide the system-oriented security engineering to: 1) establish a comprehensive understanding and translation of problem definition (see section. 4.2.1), 2) enrich the safety evaluation of such systems (see section. 4.2.3, 4.2.4). Thus, we discuss two separate future directions to resolve the challenges: 1) enrich the security evaluation and mitigation methods for MLBSS; 2) design of actionable semantic framework, to support system-oriented security engineering for MLBSS.

**5.3.1 Enrich the security evaluation and mitigation for MLBSS.** Current security evaluation and mitigation regarding attack and defence methods are mostly conducted on either separate model or over single-node the cloud services, i.e., machine-learning-as-a-service from AWS and Google (see section. 3.3, 4.2.3). Since the machine learning-based software systems could not be fully delivered with only hardware or software stack, how to consider the safety evaluation from both stacks could be more practical but is also much more challenging. Especially, the mitigation methods for hardware and software are technically different.

While the research works for different artefacts in MLBSS remain isolated and require further investigation, it is appreciated to witness the academic efforts of applying hardware and software co-design mechanism to ensure the security and privacy for on-device machine learning [150, 203]. The enrichment of the security evaluation and mitigation covering hardware and software stacks to incorporate security-by-design and other security engineering processes and tools will be of critical importance for MLBSS from the system-level view.

**5.3.2 Design of actionable semantic framework to support security engineering for MLBSS.** Recent works from [142] and [118] have reported the practical usage of machine learning models in safety-critical domains, such as the healthcare and autonomous driving systems. Other works have considered the inclusion of *hazard contribution modes* [168], *formal verification* [174], *generalisation error of model* [210], *system accountability ontology* [122] for MLBSS. While numerous efforts have been dedicated to support the MLBSS security, it is recognised that more efforts are needed.

In light of the early discussion by [10] for distributed information systems, we have seen a shift of distilling the security engineering practice with actionable framework design. Some recent works include the work from Ruiz et al. [154] for secure IT systems, the *ten security principles* in [115] and the recent *meta model* work by [194] for machine learning-based systems. In this regard, it is anticipated that future work can incorporate more meaningful and actionable semantic framework and modelling language to support the system-oriented security engineering for MLBSS, based on the identified attack modes and security practices in this paper.

## 6 CONCLUSION

Security of machine learning-based software systems is a burgeoning topic for both researchers and industry practitioners in safety-critical systems (e.g., the cyber-physical system and intrusion detection systems). The various attack modes in terms of different artefacts, i.e., the data, model, hardware and system, raise huge concerns for security development of systems. While the system security engineering for MLBSS has not been explored in a comprehensive manner, this paper investigates the state-of-the-practice of system security for the machine learning-based software systems. It is noted that, based on the surveyed works, more efforts should be shifted from the particular model and algorithms side towards the security from a holistic view, such as the discussed security-by-design for MLBSS. Although there are

several works pitching towards this direction, challenges such as the data-oriented, model-oriented, system-oriented security engineering are still open for improvements.

## ACKNOWLEDGMENT

The work has been supported by the Cyber Security Research Centre Limited whose activities are partially funded by the Australian Government's Cooperative Research Centres Programme

## REFERENCES

- [1] Shadi A Aljawarneh, Ali Alawneh, and Reem Jaradat. 2017. Cloud security engineering: Early stages of SDLC. *Future Generation Computer Systems* 74 (2017), 385–392.
- [2] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. 2019. Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 291–300.
- [3] Rob Ashmore, Radu Calinescu, and Colin Paterson. 2021. Assuring the Machine Learning Lifecycle: Desiderata, Methods, and Challenges. *ACM Comput. Surv.* 54, 5, Article 111 (2021), 39 pages.
- [4] Hala Assal and Sonia Chiasson. 2019. 'Think secure from the beginning' A Survey with Software Developers. In *Proceedings of the 2019 CHI conference on human factors in computing systems*. 1–13.
- [5] Giuseppe Ateniese, Luigi V Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. 2015. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks* 10, 3 (2015), 137–150.
- [6] Pallavi Bagga and Rahul Hans. 2017. Mobile agents system security: a systematic survey. *ACM Computing Surveys (CSUR)* 50, 5 (2017), 1–45.
- [7] Mauro Barni, Kassem Kallas, and Benedetta Tondi. 2019. A new backdoor attack in CNNs by training set corruption without label poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 101–105.
- [8] Carlo Batini, Daniela Barone, Michele Mastrella, Andrea Maurino, and Claudio Ruffini. 2007. A Framework And A Methodology For Data Quality Assessment And Monitoring.. In *ICIQ*. Citeseer, 333–346.
- [9] Kristian Beckers. 2015. *Pattern and Security Requirements: Engineering-Based Establishment of Security Standards*. Springer.
- [10] Bastian Best, Jan Jurgens, and Bashar Nuseibeh. 2007. Model-based security engineering of distributed information systems using UMLsec. In *29th International Conference on Software Engineering (ICSE'07)*. IEEE, 581–590.
- [11] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. 2019. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*. PMLR, 634–643.
- [12] Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition* 84 (2018), 317–331.
- [13] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 118–128.
- [14] Justus Bogner, Roberto Verdecchia, and Ilias Gerostathopoulos. 2021. Characterizing Technical Debt and Antipatterns in AI-Based Systems: A Systematic Mapping Study. In *4th International Conference on Technical Debt (TechDebt 2021)*. 1–10.
- [15] Nicholas Boucher, Ilia Shumailov, Ross Anderson, and Nicolas Papernot. 2021. Bad Characters: Imperceptible NLP Attacks. *arXiv preprint arXiv:2106.09898* (2021).
- [16] Azzedine Boukerche and Xiren Ma. 2021. Vision-Based Autonomous Vehicle Recognition: A New Challenge for Deep Learning-Based Systems. *ACM Comput. Surv.* 54, 4, Article 84 (2021), 37 pages.
- [17] Pierre Bourque, Richard E. Fairley, and IEEE Computer Society. 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0* (3rd ed.). IEEE Computer Society Press, Washington, DC, USA.
- [18] Houssem Ben Braiek and Foutse Khomh. 2020. On testing machine learning programs. *Journal of Systems and Software* 164 (2020), 110542.
- [19] Trevor Braun, Benjamin CM Fung, Farkhund Iqbal, and Babar Shah. 2018. Security and privacy challenges in smart cities. *Sustainable cities and society* 39 (2018), 499–507.
- [20] Fletcher J Buckley and Robert Poston. 1984. Software quality assurance. *IEEE Transactions on Software Engineering* 1 (1984), 36–41.
- [21] Umer Ahmed Butt, Muhammad Mehmood, Syed Bilal Hussain Shah, Rashid Amin, M Waqas Shaukat, Syed Mohsan Raza, Doug Young Suh, Md Piran, et al. 2020. A review of machine learning algorithms for cloud computing security. *Electronics* 9, 9 (2020), 1379.
- [22] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. 2018. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069* (2018).
- [23] Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz. 2020. Gan-leaks: A taxonomy of membership inference attacks against generative models. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*. 343–362.
- [24] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. 2019. DeepInspect: A Black-box Trojan Detection and Mitigation Framework for Deep Neural Networks.. In *IJCAI*. 4658–4664.

- [25] Kaixuan Chen, Dalin Zhang, Lina Yao, Bin Guo, Zhiwen Yu, and Yunhao Liu. 2021. Deep Learning for Sensor-Based Human Activity Recognition: Overview, Challenges, and Opportunities. *ACM Comput. Surv.* 54, 4, Article 77 (2021), 40 pages.
- [26] Zhenpeng Chen, Yanbin Cao, Yuanqiang Liu, Haoyu Wang, Tao Xie, and Xuanzhe Liu. 2020. A comprehensive study on challenges in deploying deep learning based software. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 750–762.
- [27] Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. 2021. Label-only membership inference attacks. In *International Conference on Machine Learning*. PMLR, 1964–1974.
- [28] Maria I Mera Collantes, Zahra Ghodsi, and Siddharth Garg. 2020. SafeTPU: A Verifiably Secure Hardware Accelerator for Deep Neural Networks. In *2020 IEEE 38th VLSI Test Symposium (VTS)*. IEEE, 1–6.
- [29] Jacson Rodrigues Correia-Silva, Rodrigo F Berriel, Claudine Badue, Alberto F de Souza, and Thiago Oliveira-Santos. 2018. Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data. In *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.
- [30] Gabriele Costa, Fabio Pinelli, Simone Soderi, and Gabriele Tolomei. 2021. Covert Channel Attack to Federated Learning Systems. *arXiv preprint arXiv:2104.10561* (2021).
- [31] Robby Costales, Chengzhi Mao, Raphael Norwitz, Bryan Kim, and Junfeng Yang. 2020. Live Trojan attacks on deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 796–797.
- [32] Noopur Davis. 2005. *Secure software development life cycle processes: A technology scouting report*. Technical Report. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.
- [33] Elizamary de Souza Nascimento, Iftekhar Ahmed, Edson Oliveira, Márcio Piedade Palheta, Igor Steinmacher, and Tayana Conte. 2019. Understanding Development Process of Machine Learning Systems: Challenges and Solutions. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 1–6.
- [34] Xiaoning Du, Xiaofei Xie, Yi Li, Lei Ma, Yang Liu, and Jianjun Zhao. 2019. Deepstellar: Model-based quantitative analysis of stateful deep learning systems. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 477–487.
- [35] Dick Fairley, Alice Squires, and Keith (INCOSE Systems Security Working Group 2020) Willett. 2020. System security – Guide to the Systems Engineering Body of Knowledge. [https://www.sebokwiki.org/wiki/System\\_Security](https://www.sebokwiki.org/wiki/System_Security) [Online; accessed 1-Nov-2021].
- [36] Ming Fan, Ziliang Si, Xiaofei Xie, Yang Liu, and Ting Liu. 2021. Text Backdoor Detection Using An Interpretable RNN Abstract Model. *IEEE Transactions on Information Forensics and Security* (2021).
- [37] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. 2020. Local model poisoning attacks to byzantine-robust federated learning. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*. 1605–1622.
- [38] Anna Fariha, Ashish Tiwari, Alexandra Meliou, Arjun Radhakrishna, and Sumit Gulwani. 2021. CoCo: Interactive Exploration of Conformance Constraints for Data Understanding and Data Cleaning. In *Proceedings of the 2021 International Conference on Management of Data*. 2706–2710.
- [39] Anna Fariha, Ashish Tiwari, Arjun Radhakrishna, Sumit Gulwani, and Alexandra Meliou. 2021. Conformance constraint discovery: Measuring trust in data-driven systems. In *Proceedings of the 2021 International Conference on Management of Data*. 499–512.
- [40] Michael Felderer and Rudolf Ramler. 2021. Quality Assurance for AI-based Systems: Overview and Challenges. *International Conference on Software Quality*.
- [41] Harald Foidl and Michael Felderer. 2019. Risk-based data validation in machine learning-based software systems. In *proceedings of the 3rd ACM SIGSOFT international workshop on machine learning techniques for software quality evaluation*. 13–18.
- [42] Sainyam Galhotra, Anna Fariha, Raoni Lourenço, Juliana Freire, Alexandra Meliou, and Divesh Srivastava. 2021. DataExposer: Exposing Disconnect between Data and Systems. *arXiv preprint arXiv:2105.06058* (2021).
- [43] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. 2018. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 619–633.
- [44] Lydia Gauerhof, Richard Hawkins, Chiara Picardi, Colin Paterson, Yuki Hagiwara, and Ibrahim Habli. 2020. Assuring the safety of machine learning for pedestrian detection at crossings. In *International Conference on Computer Safety, Reliability, and Security*. Springer, 197–212.
- [45] Simos Gerasimou, Hasan Ferit Eniser, Alper Sen, and Alper Cakan. 2020. Importance-driven deep learning system testing. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, 702–713.
- [46] Sukhpal Singh Gill, Minxian Xu, Carlo Ottaviani, Panos Patros, Rami Bahsoon, Arash Shaghaghi, Mohammed Golec, Vlado Stankovski, Huaming Wu, Ajith Abraham, et al. 2022. AI for next generation computing: Emerging trends and future directions. *Internet of Things* 19 (2022), 100514.
- [47] Antonio Ginart, Martin Zhang, and James Zou. 2021. MLDemon: Deployment Monitoring for Machine Learning Systems. *arXiv preprint arXiv:2104.13621* (2021).
- [48] Görkem Giray. 2021. A software engineering perspective on engineering machine learning systems: State of the art and challenges. *Journal of Systems and Software* 180 (2021), 111031.
- [49] Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. 2020. Dataset Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses. *arXiv preprint arXiv:2012.10544* (2020).
- [50] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2019. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access* 7 (2019), 47230–47244.

- [51] Qianyu Guo, Sen Chen, Xiaofei Xie, Lei Ma, Qiang Hu, Hongtao Liu, Yang Liu, Jianjun Zhao, and Xiaohong Li. 2019. An empirical study towards characterizing deep learning development and deployment across different frameworks and platforms. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 810–822.
- [52] Wenbo Guo, Lun Wang, Xinyi Xing, Min Du, and Dawn Song. 2019. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. *arXiv preprint arXiv:1908.01763* (2019).
- [53] Charles Haley, Robin Laney, Jonathan Moffett, and Bashar Nuseibeh. 2008. Security requirements engineering: A framework for representation and analysis. *IEEE Transactions on Software Engineering* 34, 1 (2008), 133–153.
- [54] Koichi Hamada, Fuyuki Ishikawa, Satoshi Masuda, Tomoyuki Myojin, Yasuharu Nishi, Hideto Ogawa, Takahiro Toku, Susumu Tokumoto, Kazunori Tsuchiya, Yasuhiro Ujita, et al. 2020. Guidelines for Quality Assurance of Machine Learning-based Artificial Intelligence.. In *SEKE*. 335–341.
- [55] Yingzhe He, Guozhu Meng, Kai Chen, Xingbo Hu, and Jinwen He. 2020. Towards Security Threats of Deep Learning Systems: A Survey. *IEEE Transactions on Software Engineering* (2020).
- [56] Bernd Heinrich, Diana Hristova, Mathias Klier, Alexander Schiller, and Michael Szwartowicz. 2018. Requirements for data quality metrics. *Journal of Data and Information Quality (JDIQ)* 9, 2 (2018), 1–32.
- [57] Sven Herpig. 2019. Securing artificial intelligence—Part 1: The attack surface of machine learning and its implications. *Think Tank at the Intersection of Technology and Society, Stiftung Neue Verantwortung, Berlin, Oct. [Online]*. Available: [https://www.stiftung-nv.de/sites/default/files/securing\\_artificial\\_intelligence.pdf](https://www.stiftung-nv.de/sites/default/files/securing_artificial_intelligence.pdf) (2019).
- [58] Hans-Martin Heyn, Eric Knauss, Amna Pir Muhammad, Olof Eriksson, Jennifer Linder, Padmini Subbiah, Shameer Kumar Pradhan, and Sagar Tungal. 2021. Requirement Engineering Challenges for AI-intense Systems Development. In *2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN) at the 43rd International Conference on Software Engineering (ICSE 2021)*. 1–8.
- [59] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. 2017. Deep models under the GAN: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 603–618.
- [60] Dorjan Hitaj, Briland Hitaj, and Luigi V Mancini. 2019. Evasion attacks against watermarking techniques found in MLaaS systems. In *2019 Sixth International Conference on Software Defined Systems (SDS)*. IEEE, 55–63.
- [61] Jennifer Horkoff. 2019. Non-functional requirements for machine learning: Challenges and new directions. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*. IEEE, 386–391.
- [62] Hongsheng Hu, Zoran Salcic, Gillian Dobbie, and Xuyun Zhang. 2022. Membership Inference Attacks on Machine Learning: A Survey. *ACM Computing Surveys (CSUR)* (2022), 1–35.
- [63] Xing Hu, Yang Zhao, Lei Deng, Ling Liang, Pengfei Zuo, Jing Ye, Yingyan Lin, and Yuan Xie. 2021. Practical attacks on deep neural networks by memory Trojaning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40, 6 (2021), 1230–1243.
- [64] Yujin Huang, Han Hu, and Chunyang Chen. 2021. Robustness of on-device Models: Adversarial Attack to Deep Learning Models on Android Apps. In *Proceedings of the 43rd International Conference on Software Engineering, Software Engineering in Practice Track*. 1–10. arXiv:cs.LG/2101.04401
- [65] Nargiz Humbatova, Gunel Jahangirova, Gabriele Bavota, Vincenzo Riccio, Andrea Stocco, and Paolo Tonella. 2020. Taxonomy of real faults in deep learning systems. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 1110–1121.
- [66] Nick Hynes, D Sculley, and Michael Terry. 2017. The data linter: Lightweight, automated sanity checking for ml data sets. In *NIPS MLsys Workshop*. 1–7.
- [67] IBM. 2021. What is Data Security? Data Security Definition and Overview. <https://www.ibm.com/topics/data-security>. [Online; accessed 06-Sep-2021].
- [68] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. 2018. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*. PMLR, 2137–2146.
- [69] ISO ISO. 2011. *26262-1-Road Vehicles Functional Safety Part 1 Vocabulary*. Technical Report. Technical report, International Organization for Standardization/Technical ....
- [70] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. 2020. High accuracy and high fidelity extraction of neural networks. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*. 1345–1362.
- [71] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. 2018. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 19–35.
- [72] Yujie Ji, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. 2018. Model-reuse attacks on deep learning systems. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 349–363.
- [73] Meenu Mary John, Helena Holmström Olsson, and Jan Bosch. 2020. Architecting AI Deployment: A Systematic Review of State-of-the-art and State-of-practice Literature. In *International Conference on Software Business*. Springer, 14–29.
- [74] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. 2021. Highly accurate protein structure prediction with AlphaFold. *Nature* (2021), 1–11.
- [75] Mika Juuti, Sebastian Szylar, Samuel Marchal, and N Asokan. 2019. PRADA: protecting against DNN model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 512–527.
- [76] Christian Kästner and Eunsuk Kang. 2020. Teaching Software Engineering for AI-Enabled Systems. In *2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 45–48.
- [77] Ashley Hyowon Kim. 2020. *The impact of platform vulnerabilities in AI systems*. Ph.D. Dissertation. Massachusetts Institute of Technology.

- [78] Jinhan Kim, Robert Feldt, and Shin Yoo. 2019. Guiding deep learning system testing using surprise adequacy. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 1039–1049.
- [79] Franz J Király, Markus Löning, Anthony Blaom, Ahmed Gueciour, and Raphael Sonabend. 2021. Designing Machine Learning Toolboxes: Concepts, Principles and Patterns. *arXiv preprint arXiv:2101.04938* (2021).
- [80] Kalpesh Krishna, Gaurav Singh Tomar, Ankur P Parikh, Nicolas Papernot, and Mohit Iyyer. 2020. Thieves on Sesame Street! Model Extraction of BERT-based APIs. In *International Conference on Learning Representations*. 1–19.
- [81] Oualid Ktata and Ghislain Lévesque. 2010. Designing and Implementing a Measurement Program for Scrum Teams: What do agile developers really need and want?. In *Proceedings of the Third C\* Conference on Computer Science and Software Engineering*. 101–107.
- [82] Fumihiro Kumeno. 2019. Software engineering challenges for machine learning applications: A literature review. *Intelligent Decision Technologies* 13, 4 (2019), 463–476.
- [83] Keita Kurita, Paul Michel, and Graham Neubig. 2020. Weight Poisoning Attacks on Pretrained Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2793–2806.
- [84] Alexander Lavin, Ciarán M Gilligan-Lee, Alessya Visnjic, Siddha Ganju, Dava Newman, Sujoy Ganguly, Danny Lange, Atılım Güneş Baydin, Amit Sharma, Adam Gibson, et al. 2021. Technology Readiness Levels for Machine Learning Systems. *arXiv preprint arXiv:2101.03989* (2021).
- [85] Alexander Lavin and Gregory Renard. 2020. Technology Readiness levels for AI & ML. In *ICML 2020 Workshop on Challenges in Deploying Machine Learning Systems*.
- [86] Younghwa Lee, Jintae Lee, and Zoonky Lee. 2002. Integrating software lifecycle process standards with security engineering. *Computers & Security* 21, 4 (2002), 345–355.
- [87] Grace A Lewis, Stephany Bellomo, and İpek Ozkaya. 2021. Characterizing and Detecting Mismatch in Machine-Learning-Enabled Systems. In *2021 IEEE/ACM 43rd International Conference on Software Engineering 1st Workshop on AI Engineering: Software Engineering for AI (WAIN 2021)*. 1–8.
- [88] Chaoran Li, Xiao Chen, Derui Wang, Sheng Wen, Muhammad Ejaz Ahmed, Seyit Camtepe, and Yang Xiang. 2021. Backdoor Attack on Machine Learning Based Android Malware Detectors. *IEEE Transactions on Dependable and Secure Computing* 01 (2021), 1–1.
- [89] Yu Li, Min Li, Bo Luo, Ye Tian, and Qiang Xu. 2020. Deepdyve: Dynamic verification for deep neural networks. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 101–112.
- [90] Yu Li, Yannan Liu, Min Li, Ye Tian, Bo Luo, and Qiang Xu. 2019. D2nn: a fine-grained dual modular redundancy framework for deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*. 138–147.
- [91] Yiming Li, Tongqing Zhai, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shutao Xia. 2020. Rethinking the trigger of backdoor attack. *arXiv preprint arXiv:2004.04692* (2020).
- [92] Bo Liu, Ming Ding, Sina Shaham, Wenny Rahayu, Farhad Farokhi, and Zihuai Lin. 2021. When machine learning meets privacy: A survey and outlook. *ACM Computing Surveys (CSUR)* 54, 2 (2021), 1–36.
- [93] Jiakun Liu, Qiao Huang, Xin Xia, Emad Shihab, David Lo, and Shanping Li. 2020. Is using deep learning frameworks free? characterizing technical debt in deep learning frameworks. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Society*. 1–10.
- [94] Tao Liu, Nuo Xu, Qi Liu, Yanzhi Wang, and Wujie Wen. 2019. A system-level perspective to understand the vulnerability of deep learning systems. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. 506–511.
- [95] Yang Liu, Lei Ma, and Jianjun Zhao. 2019. Secure deep learning engineering: A road towards quality assurance of intelligent systems. In *International Conference on Formal Engineering Methods*. Springer, 3–15.
- [96] Yuntao Liu, Ankit Mondal, Abhishek Chakraborty, Michael Zuzak, Nina Jacobsen, Daniel Xing, and Ankur Srivastava. 2020. A survey on neural trojans. In *2020 21st International Symposium on Quality Electronic Design (ISQED)*. IEEE, 33–39.
- [97] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).
- [98] Yannan Liu, Lingxiao Wei, Bo Luo, and Qiang Xu. 2017. Fault injection attack on deep neural network. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 131–138.
- [99] Yuntao Liu, Yang Xie, and Ankur Srivastava. 2017. Neural trojans. In *2017 IEEE International Conference on Computer Design (ICCD)*. IEEE, 45–48.
- [100] Zizhen Liu, Jing Ye, Xing Hu, Huawei Li, Xiaowei Li, and Yu Hu. 2020. Sequence triggered hardware trojan in neural network accelerator. In *2020 IEEE 38th VLSI Test Symposium (VTS)*. IEEE, 1–6.
- [101] Tobias Lorenz, Marta Kwiatkowska, and Mario Fritz. 2021. Backdoor Attacks on Network Certification via Data Poisoning. *arXiv preprint arXiv:2108.11299* (2021).
- [102] Lucy Ellen Lwakatare, Aiswarya Raj, Ivica Crnkovic, Jan Bosch, and Helena Holmström Olsson. 2020. Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions. *Information and Software Technology* 127 (2020), 106368.
- [103] Lucy Ellen Lwakatare, Ellinor Ränge, Ivica Crnkovic, and Jan Bosch. 2021. On the experiences of adopting automated data validation in an industrial machine learning project. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 248–257.
- [104] Lei Ma, Felix Juefei-Xu, Minhui Xue, Qiang Hu, Sen Chen, Bo Li, Yang Liu, Jianjun Zhao, Jianxiong Yin, and Simon See. 2018. Secure deep learning engineering: A software quality assurance perspective. *arXiv preprint arXiv:1810.04538* (2018).

- [105] Lei Ma, Felix Juefei-Xu, Minhui Xue, Bo Li, Li Li, Yang Liu, and Jianjun Zhao. 2019. Deepct: Tomographic combinatorial testing for deep learning systems. In *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 614–618.
- [106] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, et al. 2018. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 120–131.
- [107] Lei Ma, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Felix Juefei-Xu, Chao Xie, Li Li, Yang Liu, Jianjun Zhao, et al. 2018. Deepmutation: Mutation testing of deep learning systems. In *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 100–111.
- [108] Shiqing Ma and Yingqi Liu. 2019. Nic: Detecting adversarial samples with neural network invariant checking. In *Proceedings of the 26th Network and Distributed System Security Symposium (NDSS 2019)*.
- [109] Saeed Mahloojifar. 2019. Universal multi-party poisoning attacks. In *Proceedings of the 36th International Conference on Machine Learning*. 4274–4283.
- [110] Silverio Martínez-Fernández, Justus Bogner, Xavier Franch, Marc Oriol, Julien Siebert, Adam Trendowicz, Anna Maria Vollmer, and Stefan Wagner. 2021. Software Engineering for AI-Based Systems: A Survey. *arXiv preprint arXiv:2105.01984* (2021).
- [111] Vivien Marx. 2013. The big challenges of big data. *Nature* 498, 7453 (2013), 255–260.
- [112] Gary McGraw. 2004. Software security. *IEEE Security & Privacy* 2, 2 (2004), 80–83.
- [113] Gary McGraw, Richie Bonett, Harold Figueroa, and Victor Shepardson. 2019. Security engineering for machine learning. *Computer* 52, 8 (2019), 54–57.
- [114] Gary McGraw, Richie Bonett, Victor Shepardson, and Harold Figueroa. 2020. The Top 10 Risks of Machine Learning Security. *Computer* 53, 6 (2020), 57–61.
- [115] Gary McGraw, Harold Figueroa, Victor Shepardson, and Richie Bonett. 2020. An architectural risk analysis of machine learning systems: Toward more secure machine learning. *Berryville Institute of Machine Learning, Clarke County, VA. Accessed on: Mar 23* (2020).
- [116] Daniel Mellado, Carlos Blanco, Luis E Sánchez, and Eduardo Fernández-Medina. 2010. A systematic review of security requirements engineering. *Computer Standards & Interfaces* 32, 4 (2010), 153–165.
- [117] Sparsh Mittal, Himanshi Gupta, and Srishti Srivastava. 2021. A Survey on hardware security of DNN models and accelerators. *Journal of Systems Architecture* (2021), 102163.
- [118] Jessica Morley, Caroline Morton, Kassandra Karpathakis, Mariarosaria Taddeo, and Luciano Floridi. 2021. Towards a framework for evaluating the safety, acceptability and efficacy of AI systems for health: an initial synthesis. *arXiv preprint arXiv:2104.06910* (2021).
- [119] Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. 2021. A survey on security and privacy of federated learning. *Future Generation Computer Systems* 115 (2021), 619–640.
- [120] Mehran Mozaffari-Kermani, Susmita Sur-Kolay, Anand Raghunathan, and Niraj K Jha. 2014. Systematic poisoning attacks on and defenses for machine learning in healthcare. *IEEE journal of biomedical and health informatics* 19, 6 (2014), 1893–1905.
- [121] Hoda Naghibiyouybari, Ajaya Neupane, Zhiyun Qian, and Nael Abu-Ghazaleh. 2018. Rendered insecure: Gpu side channel attacks are practical. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 2139–2153.
- [122] Iman Naja, Milan Markovic, Peter Edwards, and Caitlin Cottrill. 2021. A Semantic Framework to Support AI System Accountability and Audit. In *European Semantic Web Conference*. Springer, 160–176.
- [123] Soroosh Nalchigar and Eric Yu. 2017. Conceptual modeling for business analytics: a framework and potential benefits. In *2017 IEEE 19th Conference on Business Informatics (CBI)*, Vol. 1. IEEE, 369–378.
- [124] Soroosh Nalchigar, Eric Yu, and Karim Keshavjee. 2021. Modeling machine learning requirements from three perspectives: a case report from the healthcare domain. *Requirements Engineering* 26, 2 (2021), 237–254.
- [125] Soroosh Nalchigar, Eric Yu, and Rajgopal Ramani. 2016. A conceptual modeling framework for business analytics. In *International Conference on Conceptual Modeling*. Springer, 35–49.
- [126] Ali Bou Nassif, Manar Abu Talib, Qassim Nasir, Halah Albadani, and Fatima Mohamad Dakalbab. 2021. Machine learning for cloud security: a systematic review. *IEEE Access* 9 (2021), 20717–20735.
- [127] Mahdi Nejadgholi and Jinqiu Yang. 2019. A study of oracle approximations in testing deep learning libraries. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 785–796.
- [128] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D Joseph, Benjamin IP Rubinstein, Udam Saini, Charles Sutton, J Doug Tygar, and Kai Xia. 2008. Exploiting machine learning to subvert your spam filter. *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats* 8 (2008), 1–9.
- [129] Florian Nuding and Rudolf Mayer. 2020. Poisoning attacks in federated learning: An evaluation on traffic sign classification. In *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*. 168–170.
- [130] Jukka K Nurminen, Tuomas Halvari, Juha Harviainen, Juha Mylläri, Antti Röyskö, Juuso Silvennoinen, and Tommi Mikkonen. 2019. Software framework for data fault injection to test machine learning systems. In *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 294–299.
- [131] Augustus Odena, Catherine Olsson, David Andersen, and Ian Goodfellow. 2019. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. In *International Conference on Machine Learning*. PMLR, 4901–4911.
- [132] Seong Joon Oh, Bernt Schiele, and Mario Fritz. 2019. Towards reverse-engineering black-box neural networks. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 121–144.

- [133] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. 2019. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4954–4963.
- [134] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. 48–53.
- [135] Andrei Paleyes, Raoul-Gabriel Urma, and Neil D Lawrence. 2020. Challenges in deploying machine learning: a survey of case studies. *arXiv preprint arXiv:2011.09926* (2020).
- [136] Nicolas Papernot. 2018. A Marauder’s map of security and privacy in machine learning: an overview of current and future research directions for making machine learning secure and private. In *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*. 1–1.
- [137] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277* (2016).
- [138] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. 506–519.
- [139] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. 2016. Towards the science of security and privacy in machine learning. *arXiv preprint arXiv:1611.03814* (2016).
- [140] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P Wellman. 2018. Sok: Security and privacy in machine learning. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 399–414.
- [141] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. Deepxplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles*. 1–18.
- [142] Zi Peng, Jingiu Yang, Tse-Hsun Chen, and Lei Ma. 2020. A first look at the integration of machine learning models in complex autonomous driving systems: a case study on Apollo. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1240–1250.
- [143] Roberto Perdisci, David Dagon, Wenke Lee, Prahlad Fogla, and Monirul Sharif. 2006. Misleading worm signature generators using deliberate noise injection. In *2006 IEEE Symposium on Security and Privacy (S&P’06)*. IEEE, 15–29.
- [144] Hung Viet Pham, Thibaud Lutellier, Weizhen Qi, and Lin Tan. 2019. CRADLE: cross-backend validation to detect and localize bugs in deep learning libraries. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 1027–1038.
- [145] Frank Piessens. 2002. A taxonomy of causes of software vulnerabilities in internet software. In *Supplementary Proceedings of the 13th International Symposium on Software Reliability Engineering*. Citeseer, 47–52.
- [146] Neoklis Polyzotis, Sudip Roy, Steven Euijong Whang, and Martin Zinkevich. 2017. Data management challenges in production machine learning. In *Proceedings of the 2017 ACM International Conference on Management of Data*. 1723–1726.
- [147] Jennifer Prendki. 2018. The curse of big data labeling and three ways to solve it.
- [148] Shirley Radack. 2009. *The system development life cycle (sdlc)*. Technical Report. National Institute of Standards and Technology.
- [149] Md Saidur Rahman, Emilio Rivera, Foutse Khomh, Yann-Gaël Guéhéneuc, and Bernd Lehnert. 2019. Machine learning software engineering in practice: An industrial case study. *arXiv preprint arXiv:1906.07154* (2019).
- [150] Francesco Regazzoni, Shivam Bhasin, Amir Ali Pou, Ihab Alshaer, Furkan Aydin, Aydin Aysu, Vincent Beroule, Giorgio Di Natale, Paul Franzon, David Hely, et al. 2020. Machine Learning and Hardware security: Challenges and Opportunities-Invited Talk. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 1–6.
- [151] Vincenzo Riccio, Gunel Jahangirova, Andrea Stocco, Nargiz Humbatova, Michael Weiss, and Paolo Tonella. 2020. Testing machine learning based systems: a systematic mapping. *Empirical Software Engineering* 25, 6 (2020), 5193–5254.
- [152] Ronald Ross, Michael McEvilley, and Janet Oren. 2018. Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems [including updates as of 1-03-2018]. <https://doi.org/10.6028/NIST.SP.800-160>
- [153] Benjamin IP Rubinstein, Blaine Nelson, Ling Huang, Anthony D Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and J Doug Tygar. 2009. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*. 1–14.
- [154] Jose Fran Ruiz, Marcos Arjona, Antonio Maña, and Carsten Rudolph. 2017. Security knowledge representation artifacts for creating secure IT systems. *Computers & Security* 64 (2017), 69–91.
- [155] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. 2019. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In *Network and Distributed Systems Security (NDSS) Symposium 2019*. 1–15.
- [156] Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, and Lora M Aroyo. 2021. “Everyone wants to do the model work, not the data work”: Data Cascades in High-Stakes AI. In *proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.
- [157] Spyridon Samonas and David Coss. 2014. The CIA strikes back: Redefining confidentiality, integrity and availability in security. *Journal of Information System Security* 10, 3 (2014).
- [158] Takami Sato, Junjie Shen, Ningfei Wang, Yunhan Jia, Xue Lin, and Qi Alfred Chen. 2021. Dirty Road Can Attack: Security of Deep Learning based Automated Lane Centering under Physical-World Attack. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*. 3309–3326.

- [159] Ari Schlesinger, Kenton P O'Hara, and Alex S Taylor. 2018. Let's talk about race: Identity, chatbots, and AI. In *Proceedings of the 2018 chi conference on human factors in computing systems*. 1–14.
- [160] Bruce Schneier. 2020. Attacking machine learning systems. *IEEE Annals of the History of Computing* 53, 05 (2020), 78–80.
- [161] Avi Schwarzschild, Micah Goldblum, Arjun Gupta, John P Dickerson, and Tom Goldstein. 2021. Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks. In *International Conference on Machine Learning*. PMLR, 9389–9398.
- [162] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in machine learning systems. *Advances in neural information processing systems* 28 (2015), 2503–2511.
- [163] SEI/CERT. 2020. CERT/CC Vulnerability Note VU#425163 - Machine learning classifiers trained via gradient descent are vulnerable to arbitrary misclassification attack. <https://kb.cert.org/vuls/id/425163>. [Online; accessed 06-Sep-2021].
- [164] Alex Serban, Koen van der Blom, Holger Hoos, and Joost Visser. 2020. Adoption and effects of software engineering best practices in machine learning. In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 1–12.
- [165] Burr Settles. 2009. Active learning literature survey. (2009).
- [166] Yi Shi, Kemal Davaslioglu, and Yalin E Sagduyu. 2020. Over-the-air membership inference attacks as privacy threats for deep learning-based wireless signal classifiers. In *Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning*. 61–66.
- [167] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18.
- [168] Colin Smith, Ewen Denney, and Ganesh Pai. 2020. Hazard Contribution Modes of Machine Learning Components. In *Proceedings of the Workshop on Artificial Intelligence Safety (SafeAI 2020), co-located with 34th AAAI Conference on Artificial Intelligence (AAAI 2020)*. 1–9.
- [169] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. 2017. Machine learning models that remember too much. In *Proceedings of the 2017 ACM SIGSAC Conference on computer and communications security*. 587–601.
- [170] Jie Song and Yeye He. 2021. Auto-Validate: Unsupervised Data Validation Using Data-Domain Patterns Inferred from Data Lakes. In *Proceedings of the 2021 International Conference on Management of Data*. 1678–1691.
- [171] Jakapan Suaboot, Adil Fahad, Zahir Tari, John Grundy, Abdun Naser Mahmood, Abdulmohsen Almalawi, Albert Y. Zomaya, and Khalil Drira. 2020. A Taxonomy of Supervised Learning for IDSs in SCADA Environments. *ACM Comput. Surv.* 53, 2, Article 40 (2020), 37 pages.
- [172] Octavian Suciu, Radu Marginean, Yigitcan Kaya, Hal Daume III, and Tudor Dumitras. 2018. When does machine learning {FAIL}? generalized transferability for evasion and poisoning attacks. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*. 1299–1316.
- [173] Lichao Sun, Yingtong Dou, Carl Yang, Ji Wang, Philip S Yu, Lifang He, and Bo Li. 2018. Adversarial attack and defense on graph data: A survey. *arXiv preprint arXiv:1812.10528* (2018).
- [174] Xiaowu Sun, Haitham Khedr, and Yasser Shoukry. 2019. Formal verification of neural network controlled autonomous systems. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*. 147–156.
- [175] Yiming Tang, Raffi Khatchadourian, Mehdi Bagherzadeh, Rhia Singh, Ajani Stewart, and Anita Raja. 2021. An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 238–250.
- [176] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. 2020. Data poisoning attacks against federated learning systems. In *European Symposium on Research in Computer Security*. Springer, 480–501.
- [177] Edith Tom, Aybüke Aurum, and Richard Vidgen. 2013. An exploration of technical debt. *Journal of Systems and Software* 86, 6 (2013), 1498–1516.
- [178] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*. 601–618.
- [179] Valerio Venceslai. 2020. *Fault-Injection and Neural Trojan Attacks on Spiking Neural Networks*. Ph.D. Dissertation. Politecnico di Torino.
- [180] Zhiyuan Wan, Xin Xia, David Lo, and Gail C Murphy. 2019. How does machine learning change software development practices? *IEEE Transactions on Software Engineering* (2019).
- [181] Binghui Wang and Neil Zhenqiang Gong. 2018. Stealing hyperparameters in machine learning. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 36–52.
- [182] Yunjuan Wang, Poorya Mianjy, and Raman Arora. 2021. Robust Learning for Data Poisoning Attacks. In *International Conference on Machine Learning*. PMLR, 10859–10869.
- [183] Zan Wang, Ming Yan, Junjie Chen, Shuang Liu, and Dongdi Zhang. 2020. Deep learning library testing via effective model generation. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 788–799.
- [184] Junyi Wei, Yicheng Zhang, Zhe Zhou, Zhou Li, and Mohammad Abdullah Al Faruque. 2020. Leaky DNN: Stealing deep-learning model secret with GPU context-switching side-channel. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 125–137.
- [185] Ying Wei, Xiaobing Sun, Lili Bo, Sicong Cao, Xin Xia, and Bin Li. [n.d.]. A comprehensive study on security bug characteristics. *Journal of Software: Evolution and Process* ([n. d.]), e2376.
- [186] Matthew Wicker, Xiaowei Huang, and Marta Kwiatkowska. 2018. Feature-guided black-box safety testing of deep neural networks. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 408–426.
- [187] Carl Wilhjelm and Awad A Younis. 2020. A Threat Analysis Methodology for Security Requirements Elicitation in Machine Learning Based Systems. In *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE, 426–433.

- [188] Oliver Willers, Sebastian Sudholt, Shervin Raafatnia, and Stephanie Abrecht. 2020. Safety concerns and mitigation approaches regarding the use of deep learning in safety-critical perception tasks. In *International Conference on Computer Safety, Reliability, and Security*. Springer, 336–350.
- [189] Peratham Wiriyathammabhum, Douglas Summers-Stay, Cornelia Fermüller, and Yiannis Aloimonos. 2016. Computer Vision and Natural Language Processing: Recent Approaches in Multimedia and Robotics. *ACM Comput. Surv.* 49, 4, Article 71 (2016), 44 pages.
- [190] Zhen Xiang, David J Miller, and George Kesisidis. 2020. Detection of backdoors in trained classifiers without access to the training set. *IEEE Transactions on Neural Networks and Learning Systems* (2020).
- [191] Yan Xiao, Ivan Beschastnikh, David S Rosenblum, Changsheng Sun, Sebastian Elbaum, Yun Lin, and Jin Song Dong. 2021. Self-Checking Deep Neural Networks in Deployment. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 372–384.
- [192] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. 2019. DeepHunter: a coverage-guided fuzz testing framework for deep neural networks. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 146–157.
- [193] Xiaofei Xie, Lei Ma, Haijun Wang, Yuekang Li, Yang Liu, and Xiaohong Li. 2019. DiffChaser: Detecting Disagreements for Deep Neural Networks.. In *IJCAI*. 5772–5778.
- [194] Pulei Xiong, Scott Buffett, Shahrear Iqbal, Philippe Lamontagne, Mohammad Mamun, and Heather Molyneaux. 2021. Towards a Robust and Trustworthy Machine Learning System Development. *arXiv preprint arXiv:2101.03042* (2021).
- [195] Qian Xu, Md Tanvir Arafin, and Gang Qu. 2021. Security of Neural Networks from Hardware Perspective: A Survey and Beyond. In *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 449–454.
- [196] Mingfu Xue, Chongyan Gu, Weiqiang Liu, Shichao Yu, and Máire O'Neill. 2020. Ten years of hardware Trojans: a survey from the attacker's perspective. *IET Computers & Digital Techniques* 14, 6 (2020), 231–246.
- [197] Chen Yang, Jia Hou, Minshun Wu, Kuizhi Mei, and Li Geng. 2020. Hardware Trojan Attacks on the Reconfigurable Interconnections of Convolutional Neural Networks Accelerators. In *2020 IEEE 15th International Conference on Solid-State & Integrated Circuit Technology (ICSICT)*. IEEE, 1–3.
- [198] Pan Yang, Naixue Xiong, and Jingli Ren. 2020. Data security and privacy protection for cloud storage: A survey. *IEEE Access* 8 (2020), 131723–131740.
- [199] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 1–19.
- [200] Zhibin Yang, Yang Bao, Yongqiang Yang, Zhiqiu Huang, Jean-Paul Bodeveix, Mamoun Filali, and Zonghua Gu. 2021. Exploiting augmented intelligence in the modeling of safety-critical autonomous systems. *Formal Aspects of Computing* (2021), 1–42.
- [201] Jing Ye, Yu Hu, and Xiaowei Li. 2018. Hardware Trojan in FPGA CNN accelerator. In *2018 IEEE 27th Asian Test Symposium (ATS)*. IEEE, 68–73.
- [202] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. 2020. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering* (2020), 1–37.
- [203] Rui Zhang, Mingcong Song, Tao Li, Zhibin Yu, Yuting Dai, Xiaoguang Liu, and Gang Wang. 2021. Democratic learning: A hardware/software co-design for lightweight blockchain-secured on-device machine learning. *Journal of Systems Architecture* (2021), 102205.
- [204] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.
- [205] Tianyi Zhang, Cuiyun Gao, Lei Ma, Michael Lyu, and Miryung Kim. 2019. An empirical study of common challenges in developing deep learning applications. In *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 104–115.
- [206] Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)* 11, 3 (2020), 1–41.
- [207] Xiyue Zhang, Xiaofei Xie, Lei Ma, Xiaoning Du, Qiang Hu, Yang Liu, Jianjun Zhao, and Meng Sun. 2020. Towards characterizing adversarial defects of deep learning software from the lens of uncertainty. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, 739–751.
- [208] Xufan Zhang, Yilin Yang, Yang Feng, and Zhenyu Chen. 2019. Software engineering practice in the development of deep learning applications. *arXiv preprint arXiv:1910.03156* (2019).
- [209] Yuhao Zhang, Yifan Chen, Shing-Chi Cheung, Yingfei Xiong, and Lu Zhang. 2018. An empirical study on TensorFlow program bugs. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 129–140.
- [210] Xingyu Zhao, Alex Banks, James Sharp, Valentin Robu, David Flynn, Michael Fisher, and Xiaowei Huang. 2020. A safety framework for critical systems utilising deep neural networks. In *International Conference on Computer Safety, Reliability, and Security*. Springer, 244–259.
- [211] Xiao-Ping Zhao and Rui Jiang. 2020. Distributed machine learning oriented data integrity verification scheme in cloud computing environment. *IEEE Access* 8 (2020), 26372–26384.
- [212] Liehuang Zhu, Yulu Wu, Keke Gai, and Kim-Kwang Raymond Choo. 2019. Controllable and trustworthy blockchain-based cloud data management. *Future Generation Computer Systems* 91 (2019), 527–535.
- [213] Yuankun Zhu, Yueqiang Cheng, Husheng Zhou, and Yantao Lu. 2021. Hermes Attack: Steal {DNN} Models with Lossless Inference Accuracy. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*.