Adversarial Machine Learning - Industry Perspectives

Ram Shankar Siva Kumar*, Magnus Nyström†, John Lambert‡, Andrew Marshall§, Mario Goertzel¶
, Andi Comissoneru‖, Matt Swann** and Sharon Xia††
Microsoft
Redmond,USA
Email: *Ram.Shankar@microsoft.com, †mnystrom@microsoft.com, ‡johnla@microsoft.com, §amarshal@microsoft.com
¶mariogo@microsoft.com, ‖andic@microsoft.com, **mswann@microsoft.com,
††shxia@microsoft.com

***Abstract*—Based on interviews with 28 organizations, we found that industry practitioners are not equipped with tactical and strategic tools to protect, detect and respond to attacks on their Machine Learning (ML) systems. We leverage the insights from the interviews and enumerate the gaps in securing machine learning systems when viewed in the context of traditional software security development. We write this paper from the perspective of two personas: developers/ML engineers and security incident responders. The goal of this paper is to layout the research agenda to amend the Security Development Lifecycle for industrial-grade software in the adversarial ML era.**

***Index Terms*—adversarial machine learning, software security, engineering**

I. Introduction

Adversarial Machine Learning is now having a moment in the software industry - For instance, Google [1], Microsoft [2] and IBM [3] have signaled, separate from their commitment to securing their traditional software systems, initiatives to secure ML systems. In Feb 2019, Gartner, the leading industry market research firm, published its first report on adversarial machine learning [4] advising that *Application leaders must anticipate and prepare to mitigate potential risks of data corruption, model theft, and adversarial samples.* The motivation for this paper is to understand the extent to which organizations across different industries are protecting their ML systems from attacks, detecting adversarial manipulation and to responding to attacks on their ML systems.

There are many reasons why organizations may already be ahead of the curve in systematically securing their ML assets. Firstly, in the last three years, companies heavily investing in machine learning themselves - Google , Amazon , Microsoft , Tesla faced some degree of adversarial attacks [5]–[8]; a bellwether of the rise of adversarial machine learning. Secondly, standards organizations like ISO [9] are forming certification rubrics to assess security of ML systems and whose endorsements have been historically sought after in the industry [10]. Also, governments are showing signs that industry will have to build ML systems securely, with the European Union even releasing a complete checklist to assess trustworthiness of ML systems [11] Finally, ML is rapidly becoming core to organizations' value proposition (with a projected Annual Growth Rate of 39% for machine learning investments in 2020 [12]) and it is only natural that organizations invest in protecting their crown jewels.

We make two contributions in this paper:

1) Despite the compelling reasons to secure ML systems, over a survey spanning 28 different organizations, we found that most industry practitioners are yet to come to terms with adversarial machine learning. 25 out of the 28 organizations indicated that they dont have the right tools in place to secure their ML systems and are explicitly looking for guidance.
2) We enumerate the security engineering aspects of building ML systems using Security development Lifecycle (SDL) frame work, the de facto software building process in industry.

This paper is a compendium of pain points and gaps in securing machine learning systems as encountered by typical software organizations. We hope to appeal to the research community to help solve the problem faced by two personas - software developers/ML engineers and security incident responders - when securing machine learning systems. The goal of this paper is to engage ML researchers to revise and amend Security Development Lifecycle for industrial-grade software in the adversarial ML era.

The paper is organized thus: the first part outlines the survey methodology and findings. The second part comprises gaps in securing machine learning in three phases: when ML systems are designed and developed; when the said system is prepped for deployment and it is under attack.

II. Industry Survey about Adversarial ML

We interviewed 28 organizations spanning Fortune 500, small-and-medium businesses, non-profits, and government organizations to understand how they secure their machine learning systems from adversarial attacks (See Table I and Table II).

22 out of the 28, were in security sensitive fields such as finance, consulting, cybersecurity, healthcare, government. The other 6 organizations represented social media analytics, publishing, agriculture, urban planning, food processing and translation services (See Table II for distribution).

TABLE I
ORGANIZATION SIZE

Organization size	*Count*
Large Organizations (> 1000 employees)	18
Small-and-Medium Size Businesses	10

TABLE II
ORGANIZATION TYPES

Organization	*Count*
Cybersecurity	10
Healthcare	5
Government	4
Consulting	2
Banking	2
Social Media Analytics	1
Publishing	1
Agriculture	1
Urban Planning	1
Food Processing	1
Translation	1

At each organization, we interviewed two personas: the developer in charge of building machine models in the organization, and the security personnel who was on point for securing the organizations infrastructure. Depending on the size of the organization, these two personas were either in different teams, the same team or even the same person. All organizations we spoke to were familiar with the Security Development Lifecycle as pertaining to traditional software engineering, though the degree to which they executed varied larger corporations that had a more formal, documented process than small and medium sized corporations. We also limited to organizations had relatively mature machine learning investments, with a few of them centering their business around AI.

These organizations executed on their ML strategy in a variety of ways: most of them used ML toolkits such a Keras, TensorFlow or PyTorch to build ML models; 10 organizations relied on Machine Learning as a Service such as Microsofts Cognitive API [13], Amazon AI Services [14], Google CloudAI [15]; Only 2 organizations built ML systems from scratch and not relying on either existing toolkits/ML platforms (See Table III)

TABLE III
ML STRATEGY

How do you build ML Systems	*Count*
Using ML Frameworks	16
Using ML as a Service	10
Building ML Systems from scratch	2

Limitations of Study: Our sample size of 28 may not represent the entire population industries employing machine learning. For instance, the study does not include startups and has a pre-ponderance of security-sensitive organizations. We also do not account for geographic distribution most of the organizations operate and head quartered in the United States or Europe. We limited ourselves to failures that are caused by a malicious attacker in the system and did not investigate broader safety failures such as common corruption [16], reward hacking [17], distributional shifts [18] or naturally occurring adversarial examples [19].

A. Findings:

1) Though, all 28 organizations indicated that security of AI system is important to their business productivity, the emphasis is still on traditional security. As one security analyst put it, *Our top threat vector is spearphishing and malware on the box. This [adversarial ML] looks futuristic*. While there is great interest in adversarial machine learning, only 6 organizations (all of whom are large organizations or government) are ready to assign head-count to solve the problem
2) Lack of adversarial ML know-how: Organizations seem lack the tactical knowledge to secure machine learning systems in production. As one of them put it, *Traditional software attacks are a known unknown. Attacks on our ML models are unknown unknown*. 22 out of the 25 (3 government organizations abstained from answering this question satisfactorily) organizations said that they dont have the right tools in place to secure their ML systems and are explicitly looking for guidance. Also, security engineers mostly do not have the ability to detect and respond to attacks on ML systems (See Table IV)

TABLE IV
STATE OF ADVERSARIAL ML

Do you secure your ML systems today	*Count*
Yes	3
No	22

3) We walked through the list of attacks as outlined in [20] and asked them to pick the top attack that would affect their businesses(See Table V). Note: respondents were allowed to pick only one threat as opposed to stack rank them all. The result were as follows:

TABLE V
TOP ATTACK

Which attack would affect your org the most?	*Distribution*
Poisoning (e.g: [21])	10
Model Stealing (e.g: [22])	6
Model Inversion (e.g: [23])	4
Backdoored ML (e.g: [24])	4
Membership Inference (e.g: [25])	3
Adversarial Examples (e.g: [26])	2
Reprogramming ML System (e.g: [27])	0
Adversarial Example in Physical Domain (e.g: [5])	0
Malicious ML provider recovering training data (e.g: [28])	0
Attacking the ML supply chain (e.g: [24])	0
Exploit Software Dependencies (e.g: [29])	0

- Data poisoning has caught the attention of enterprises, perhaps because of the cultural significance

of Tay. A medium sized financial tech put it thus, *We use ML systems to suggest tips and financial products for our users. The integrity of our ML system matters a lot. Worried about inappropriate recommendation like attack on Tay*

- Organizations care most about attacks that can lead to potential breach of privacy. As one of the banks put it, *Want to protect client info, employee info used in ML models but we dont know have a plan in place*
- Model Stealing that can lead to loss of Intellectual property is another concern. A large retail organization said, *We run a proprietary algorithm to solve our problem and it would be worrisome if someone can reverse engineer it*
- Adversarial Examples in the physical domain, resonated with the respondents, but did not rank high on the list. One reason may be that the organizations we spoke to did not have physical component like cars or drones.

4) For security analysts, there is a mismatch between expectations and reality when it comes to adversarial ML. Many security analysts expect that algorithms available in platforms such as Keras, TensorFlow or PyTorch are inherently secure against adversarial manipulations and have already been battle tested against adversarial ML attacks. This is perhaps, because security analysts who have mostly been exposed to traditional software, assume that libraries put out by large organizations such as Facebook or Google would have been already been security stress tested. Similarly, organizations seem to push the security responsibility upstream to service providers as one of the respondents said, *We use Machine Learning as a Service and expect them to provide these robust algorithms and platforms*
5) Finally, security analysts and developers do not know what to expect when systems get attacked. As one of the ML engineers put it, *I dont expect any system to be immune from spoofing, but I need to know confidence levels and expected performance; as well as potential failure modes. If system is spoofed, what is the worst possible outcome?*

In the following sections of the paper, summarized in Fig.1 , we elaborate the gaps in current SDL process when building ML systems, as they are prepped for deployment and when the ML system is under attack. For each gap, we outline existing methods in traditional software development and sketch future research agenda.

III. About SDL

In July 2001, Microsoft was affected by CodeRed, a computer worm that affected Internet Information Server (IIS) 4.0 and 5.0 [30]. This happened because of a single line error in code running by default in IIS4 and IIS5 systems, enabling a buffer overflow attack. In Jan 2002, Microsoft halted developing any new software for 2 months to fix all known security bugs in its system, pairing security experts with developers. Out of this close interaction, a systematic process of providing security guidance evolved, helping engineers look for software defects and implementation flaws. This set of practices has now come to be called the Secure Development Lifecycle (SDL). While SDL does not eliminate all software bugs, they do help to catch software vulnerabilities that could later be exploited, before it reaches the hands of a customer. For instance, after SDL was introduced in Microsoft, the number of reported vulnerabilities between Windows XP and Windows Vista, reduced by 45%, and number of vulnerabilities between SQL Server 2000 and SQL Server 2005, reduced by 91% [31]. Currently SDL, in some form, is a de-facto process in industry-grade software development adopted by 122 organizations [32], including Google [33], IBM [34], Facebook [35] and Netflix [36].

The primary inquiry is amending and revising the SDL process used in securing traditional software, to secure ML systems against adversarial attacks.

IV. Gaps during Development of ML Solution

A. Curated repository of attacks

In traditional software security, attacks are decomposed into shareable tactics and procedures and are collectively organized in the MITRE ATT&CK framework [37]. This provides a search-able attack repository comprising, attacks by researchers as well as nation state attackers. For every attack, there is a description of the technique, which advanced persistent threat is known to use it, detection ideas as well as reference to publications with further context.

In adversarial ML, the scholarship is booming [38] but the awareness is low among developers and security analysts only 5 out of 28 organizations stated that they had working knowledge of adversarial ML. We propose that a similar curated repository of attacks be created, preferably by extending the widely used existing MITRE Framework. For instance, when adversarial ML researchers publish a new type of attack, we ask them to register their attacks in the MITRE framework, so that security analysts have a unified view of traditional and adversarial ML attacks.

B. Adversarial ML specific secure coding practices:

In traditional software setting, secure coding practice enables engineers to reduce exploitable vulnerabilities in their programs and enables auditing of source code by other engineers. For instance Python [39], Java, C and C++ [40] have well defined secure coding practice against traditional software bugs like memory corruption. In machine learning setting, adversarial ML specific security guidance is sparse. Most toolkits provide best practices (TensorFlow [41] , Pytorch [42], Keras [43]) but TensorFlow is the only framework that provides consolidated guidance around traditional software attacks [44] and links to tools for testing against adversarial attacks [45].

We think future work in adversarial ML should focus on providing best practices to eliminate undefined program

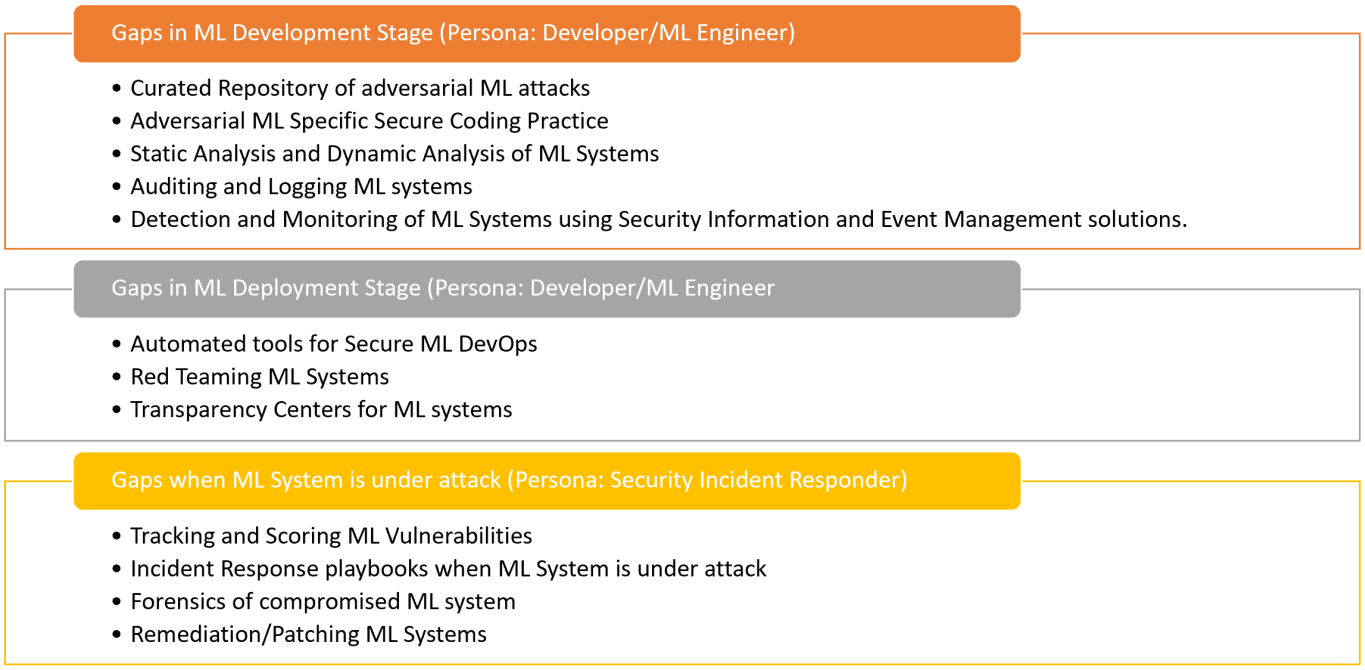

Fig. 1. Security Engineering aspects of Machine Learning

behaviors and exploitable vulnerabilities. We acknowledge it is difficult to provide concrete guidance because the field is protean [46]. Perhaps one direction, would be to enumerate guidance based on security consequence. Viewing the world through SDL allows for imperfect solutions to exist. For instance, in traditional software security, the outdated *cryptgenradnom* [47] function should not be used to generate random seeds for secret sharing protocols which are of higher security consequence), but can be used to generate process IDs in an operating system (which is of lower security consequence). Instead of thinking of secure coding practice as underwriting a strong security guarantee, a good start would be to provide examples of security-compliant and non-compliant code examples.

C. Static Analysis and Dynamic Analysis of ML Systems

In traditional software security, static analysis tools help detect potential bugs in the code without the need for execution and to detect violations in coding practices. The source code is generally converted into an abstract syntax tree, which is then used to create a control flow graph. Coding practices and checks, which are turned into logic, are searched over the control flow graph, and are raised as errors when inconsitent with logic. In traditional software for instance, in Python tools like Pyt [48] detect traditional software security vulnerabilities. Dynamic analysis, on the other hand, involves searching for vulnerabilities on executing a certain code path.

On the ML front, tools like cleverhans [45], secml [49], and the adversarial robustness toolkit [50] providing a certain degree of white-box style and blackbox style dynamic testing. A future area of research is how to extend the analysis to model stealing, model inversion and membership inference style attacks. Out of the box static analysis for adversarial ML is less explored. One promising angle is work like Code2graph [51] that generates call graphs in ML platform and in conjunction with symbolic execution may provide the first step towards a static analysis tool. We hope that the static analysis tools ultimately integrate with IDE (integrated development environment) to provide analytical insight into the syntax, semantics, so as to prevent the introduction of security vulnerabilities before the application code is committed to the code repository.

D. Auditing and Logging in ML Systems

To use a traditional software example, important security events in the operating system like process creation are logged in the host, which is then forwarded to Security Information and Event Management (SIEM) systems. This later enables, security responders to run anomaly detection [52], [53] to detect if an anomalous process (which is an indication of malware) was executed on the machine.

Auditing in ML systems was initially pointed by Papernot [54] with solution sketches to instrument ML environments to capture telemetry. As done in traditional software security, we recommend that developers of ML systems, identify high impact activities in their system. We recommend executing the list of attacks that are considered harmful to the organization and ensuring that the events manifesting in the telemetry can be traced back to the attack. Finally, these events must be exportable to traditional Security Information and Event Management systems, so that analysts can keep an audit trail for future investigations.

E. Detection and Monitoring of ML systems

Currently, ML environments are *illegible* to security analysts as they have no operational insights. There has been insightful working pointing to the brittleness of current adversarial detection mechanisms [46] and how to make them better [19]. In addition, we propose that detection methods are written so that they are easily shared among security analysts. For instance, in traditional software security, detection logic is written in a common format, the most popular of which is Sigma [55]. Where MITRE ATT&CK provides a great repository of insight in techniques used by adversaries, Sigma can turn one analyst's insights into defensive action for many, by providing a way to self-documented concrete logic for detecting an attacker's techniques.

V. Gaps when Preparing for Deployment of ML System

A. Automating Tools in Deployment Pipeline

In a typical traditional software setting, after a developer as the developer completes small chunks of the assigned task, the following sequence of steps generally follow: first, the code is committed to source control and Continuous Integration triggers application build and unit tests; once these are passed, Continuous Deployment triggers an automated deployment into testing and then production wherein it reaches the customer. At each step of the build, security tools are integrated.

We hope that dynamic analysis tools built for adversarial ML are integrated into the continuous integration / continuous delivery pipeline. Automating the adversarial ML testing, will help fix issues and without overloading engineers with too many tools or alien processes outside of their everyday engineering experience.

B. Red Teaming ML Systems

Informally, the risk of an attack to an organization depends on two factors: the impact it has on the business and the likelihood of the attack occurring. Threat modeling of ML Systems [56], performed by the ML developers, address the impact factor. Red teaming, the deliberate process of exploiting the system through any means possible conducted by an independent security team, helps to assess the likelihood factor. For critical security applications, red teaming is industry standard and a requirement for providing software to US governments [57]. With Facebook being the first industry to start an AI Red Team [58] and is unexplored area in the adversarial ML field for others.

C. Transparency Centers

In traditional security, large organizations such as Kaspersky [59], Huawei [60] have provided transparency centers where participants visit a secure facility to conduct deep levels of source code inspection and analysis. Participants would have access to source code and an environment for in-depth inspection with diagnostic tools to verify the security aspects of different products such as SSL and TCP/IP implementation or pseudorandom number generators.

In adversarial ML context, future transparency centers may need to attest over 3 modalities: that the ML platform is implemented in a secure fashion; that the MLaaS is implemented meeting basic security objectives and finally, that the ML model embedded in an edge device (such as models on mobile phones, for instance) meets basic security objectives. An interesting direction for future research is to providing tools/test harnesses to advance security assurance of products building on top of formal verification such as [61], [62] to extend to large scale ML models used in industry.

VI. Gaps when an ML System is under attack

A. Tracking and Scoring ML Vulnerabilities

In traditional software security, when a researcher finds a vulnerability in a system, it is first assigned a unique identification number and registered in a database called Common Vulnerabilities and Exposure [63]. Accompanying these vulnerabilities are severity ratings calculated by using Common Vulnerability Scoring System [64]. For instance, in the recent zero day found against Internet Explorer that allowed for remote code execution [65] the vulnerability was referred to as "CVE-2020-0674" and had assigned a base CVSS score 7.5 out of 10 [66], roughly indicating the seriousness of the bug. This enables the entire industry to refer to the problem using the same tongue.

In an ML context, we ask the adversarial ML research community to register vulnerabilities (especially affecting large groups of consumers) in a trackable system like CVE to ensure that industry manufacturers are alerted. It is not clear how ML vulnerabilities should be scored accounting for risk and impact. Finally, When a security analyst sees news about an attack, the bottom line is mostly Is my organization affected by the attack? and today, organizations lack the ability to scan an ML environment for known adversarial ML specific vulnerabilities.

B. Incident Response

When a security engineer receives a notification that an ML system is under attack, and triages that the attack is relevant to the business, there are two important steps ascertaining blast radius and preparing for containment. For instance, in the case of ransomware, a traditional software attack, the blast radius would be to determine other machines connected to the infected machine, and containment would be to remove the machines from the network for forensic analysis.

Both steps are difficult, because ML systems are highly integrated in a production setting where a failure of one can lead to unintended consequences [67]. One interesting line of research is to identify whether, if it is possible to container-ize ML systems so as to quarantine uncompromised ML systems from the impact of a compromised ML system, just as anti virus systems would quarantine an infected file.

C. Forensics

In traditional software security, once the machine is contained, it is prepared for forensics to ascertain root cause. There are a lot of open questions in this area so as to meaningfully interrogate ML systems under attack to ascertain the root cause of failure:

1) What are the artifacts that should be analyzed for every ML attack? Model file? The queries that were scored? Training data? Architecture? Telemetry? Hardware? All the software applications running on the attacked system? How can we leverage work data provenance and model provenance for forensics?
2) How should these artifacts be collected? For instance, for ML models developed on the end point or Internet of Things vs. organizations using ML as a Service, the artifacts available for analysis and acquisition methodology will be different. We posit that ML forensics methodology is dependent on ML frameworks (like PyTorch vs. TensorFlow), ML paradigms (e.g: reinforcement learning vs. supervised learning) and ML environment (running on host vs cloud vs edge).
3) An orthogonal step that may be carried out is cyberthreat attribution, wherein the security analyst is able to determine the actor responsible for the attack. In traditional software, this is done by analyzing the forensic evidence such as infrastructure used to mount the attack, threat intelligence and ascertaining the attackers tools, tactics and procedures using established rubrics called analytic trade craft [68]. It is unclear how this would be amended in the adversarial ML age.

D. Remediation

In traditional software security. Tuesday is often synonymous with Patch Tuesday. This is when companies like Microsoft, SAS, and Adobe release patches for vulnerabilities in their software, which are then installed based on an organization's patching policy.

In an ML context, when Tay was compromised because of poisoning attack, it was suspended by Microsoft. This may not be possible for all ML systems, especially those that have been deployed on the edge. It is not clear what the guidelines are for patching a system, that is vulnerable to model . On the same lines, it is not clear how one would validate if the patched ML model will perform as well as the previous one, but not be subject to the same vulnerabilities based on Papernot et. als [69] transferability result.

VII. Conclusion

In a keynote in 2019, Nicholas Carlini [70] likened the adversarial ML field to crypto pre-Shannon based on the ease with which defenses are broken. We extend Carlinis metaphor beyond just attacks and defenses: through interviews spanning 28 organizations, we found that most ML engineers and incident responders are unequipped to secure industry-grade ML systems against adversarial attacks. We also enumerate how researchers can contribute to Security Development Lifecyle (SDL), the de facto process for building reliable software, in the era of adversarial ML. We conclude that if ML is Software 2.0 [71], it also needs to follow fundamental security rigor from traditional software 1.0 development process.

VIII. Acknowledgement

We would like to thank the following Microsoft engineers for their support: Hyrum Anderson, Steve Dispensa, Avi Ben-Menahem, Seetharaman Harikrishnan, Anil Thomas, Efim Hudis, Jarek Stanley, Jeffrey Snover, Cristin Goodwin, Kevin Scott, Kymberlee Price, Mark Russinovich, Faraz Fadavi, Walner Dort, Steve Mott, Krishna Sagar B V and members of the AETHER Security Engineering group. We also would like to thank Nicolas Papernot (Google Brain) and Justin Gilmer (Google Brain), Miles Brundage (OpenAI), Ivan Evtimov (University of Washington), Frank Nagle (Harvard University); Kendra Albert (Harvard Law), Jonathon W. Penney (Citizen Lab) and Bruce Schneier (Harvard Kennedy School), Steve Lipner (SAFECode.org), Gary McGraw (BIML), Fernando Montenegro and members of AI Safety and Security Working Group at Berkman Klein Center for Internet and Society for the fruitful discussions. We would also like to thank ML engineers and security analysts from 28 organizations for their time and insights.

References

[1] "Responsible AI Practices." [Online]. Available: https://ai.google/responsibilities/responsible-ai-practices/?category=security
[2] "Securing the Future of AI and ML at Microsoft." [Online]. Available: https://docs.microsoft.com/en-us/security/securing-artificial-intelligence-machine-learning
[3] "Adversarial Machine Learning," Jul 2016. [Online]. Available: https://ibm.co/36fhajg
[4] S. A. Gartner Inc, "Anticipate Data Manipulation Security Risks to AI Pipelines." [Online]. Available: https://www.gartner.com/doc/3899783
[5] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," *arXiv preprint arXiv:1707.07397*, 2017.
[6] J. Li, S. Qu, X. Li, J. Szurley, J. Z. Kolter, and F. Metze, "Adversarial Music: Real World Audio Adversary Against Wake-word Detection System," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 908–11 918.
[7] P. L. Microsoft, "Learning from Tay's introduction," Mar 2016. [Online]. Available: https://blogs.microsoft.com/blog/2016/03/25/learning-tays-introduction/
[8] "Experimental Security Research of Tesla Autopilot," Tech. Rep. [Online]. Available: https://bit.ly/37oGdla
[9] "ISO/IEC JTC 1/SC 42 Artificial Intelligence," Jan 2019. [Online]. Available: https://www.iso.org/committee/6794475.html
[10] R. Von Solms, "Information security management: why standards are important," *Information Management & Computer Security*, vol. 7, no. 1, pp. 50–58, 1999.
[11] "Ethics guidelines for trustworthy ai," Nov 2019. [Online]. Available: https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai
[12] "2018 AI predictions 8 insights to shape business strategy," Tech. Rep. [Online]. Available: https://www.pwc.com/us/en/advisory-services/assets/ai-predictions-2018-report.pdf
[13] [Online]. Available: https://azure.microsoft.com/en-us/services/cognitive-services/
[14] [Online]. Available: https://aws.amazon.com/machine-learning/ai-services/
[15] [Online]. Available: https://cloud.google.com/products/ai/
[16] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *arXiv preprint arXiv:1903.12261*, 2019.

[17] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[18] J. Leike, M. Martic, V. Krakovna, P. A. Ortega, T. Everitt, A. Lefrancq, L. Orseau, and S. Legg, "Ai safety gridworlds," *arXiv preprint arXiv:1711.09883*, 2017.

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

[20] R. S. S. Kumar, D. O. Brien, K. Albert, S. Viljöen, and J. Snover, "Failure modes in machine learning systems," *arXiv preprint arXiv:1911.11034*, 2019.

[21] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 19–35.

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

[23] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1322–1333.

[24] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.

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

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

[27] G. F. Elsayed, I. Goodfellow, and J. Sohl-Dickstein, "Adversarial reprogramming of neural networks," *arXiv preprint arXiv:1806.11146*, 2018.

[28] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Adversarial generative nets: Neural network attacks on state-of-the-art face recognition," *arXiv preprint arXiv:1801.00349*, pp. 1556–6013, 2017.

[29] Q. Xiao, K. Li, D. Zhang, and W. Xu, "Security risks in deep learning implementations," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 123–128.

[30] C. C. Zou, W. Gong, and D. Towsley, "Code red worm propagation modeling and analysis," in *Proceedings of the 9th ACM conference on Computer and communications security*. ACM, 2002, pp. 138–147.

[31] [Online]. Available: https://bit.ly/2G4NaMv

[32] [Online]. Available: https://www.bsimm.com/

[33] [Online]. Available: https://cloud.google.com/security/overview/whitepaper

[34] [Online]. Available: https://www.ibm.com/security/secure-engineering/

[35] [Online]. Available: https://about.fb.com/news/2019/01/designing-security-for-billions/

[36] [Online]. Available: https://medium.com/@NetflixTechBlog/scaling-appsec-at-netflix-6a13d7ab6043

[37] [Online]. Available: https://about.fb.com/news/2019/01/designing-security-for-billions/

[38] N. Carlini. [Online]. Available: https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html

[39] [Online]. Available: http://www.pythonsecurity.org/

[40] [Online]. Available: https://wiki.sei.cmu.edu/confluence/display/seccode

[41] [Online]. Available: https://bit.ly/2RDl3cm

[42] [Online]. Available: https://pytorch.org/docs/stable/notes/multiprocessing.html

[43] [Online]. Available: https://keras.io/why-use-keras/

[44] [Online]. Available: https://github.com/tensorflow/tensorflow/blob/master/SECURITY.md

[45] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, A. Matyasko, V. Behzadan, K. Hambardzumyan, Z. Zhang, Y.-L. Juang, Z. Li, R. Sheatsley, A. Garg, J. Uesato, W. Gierke, Y. Dong, D. Berthelot, P. Hendricks, J. Rauber, and R. Long, "Technical report on the cleverhans v2.1.0 adversarial examples library," *arXiv preprint arXiv:1610.00768*, 2018.

[46] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 2017, pp. 3–14.

[47] [Online]. Available: hhttps://docs.microsoft.com/en-us/windows/win32/api/wincrypt/nf-wincrypt-cryptgenrandom

[48] [Online]. Available: https://github.com/python-security/pyt

[49] M. Melis, A. Demontis, M. Pintor, A. Sotgiu, and B. Biggio, "secml: A Python Library for Secure and Explainable Machine Learning," *arXiv preprint arXiv:1912.10013*, 2019.

[50] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. Molloy, and B. Edwards, "Adversarial Robustness Toolbox v1.1.0," *CoRR*, vol. 1807.01069, 2018. [Online]. Available: https://arxiv.org/pdf/1807.01069

[51] G. Gharibi, R. Tripathi, and Y. Lee, "Code2graph automatic generation of static call graphs for python source code," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. ACM, 2018, pp. 880–883.

[52] J. Twycross, U. Aickelin, and A. Whitbrook, "Detecting anomalous process behaviour using second generation artificial immune systems," *arXiv preprint arXiv:1006.3654*, 2010.

[53] W. M. Van der Aalst and A. K. A. de Medeiros, "Process mining and security: Detecting anomalous process executions and checking process conformance," *Electronic Notes in Theoretical Computer Science*, vol. 121, pp. 3–21, 2005.

[54] N. Papernot, "A marauder's map of security and privacy in machine learning," *arXiv preprint arXiv:1811.01134*, 2018.

[55] F. Roth, "Sigma." [Online]. Available: https://github.com/Neo23x0/sigma

[56] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," *arXiv preprint arXiv:1611.03814*, 2016.

[57] "Nvd." [Online]. Available: https://nvd.nist.gov/800-53/Rev4/control/CA-8

[58] B. Dolhansky, R. Howes, B. Pflaum, N. Baram, and C. C. Ferrer, "The deepfake detection challenge (dfdc) preview dataset," *arXiv preprint arXiv:1910.08854*, 2019.

[59] [Online]. Available: https://bit.ly/2v89frf

[60] [Online]. Available: https://www.huawei.com/en/about-huawei/trust-center/transparency/huawei-cyber-security-transparency-centre-brochure

[61] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient Smt solver for verifying deep neural networks," in *International Conference on Computer Aided Verification*. Springer, 2017, pp. 97–117.

[62] T.-W. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, D. Boning, I. S. Dhillon, and L. Daniel, "Towards fast computation of certified robustness for relu networks," *arXiv preprint arXiv:1804.09699*, 2018.

[63] "Common Vulnerabilities and Exposures (CVE)." [Online]. Available: https://cve.mitre.org/

[64] "Common Vulnerability Scoring System (CVSS)." [Online]. Available: https://www.first.org/cvss/specification-document

[65] [Online]. Available: https://portal.msrc.microsoft.com/en-us/security-guidance/advisory/ADV200001

[66] "CVE-2020-0674." [Online]. Available: https://kb.cert.org/vuls/id/338824/

[67] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, and M. Young, "Machine Learning: The High interest Credit Card of Technical Debt," in *SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop)*, 2014.

[68] "A Guide to Cyber Attribution," 2018. [Online]. Available: https://bit.ly/2G50UXB

[69] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[70] [Online]. Available: https://youtu.be/-p2il-V-0fk?t=1574

[71] "Software 2.0," 2017. [Online]. Available: https://medium.com/@karpathy/software-2-0-a64152b37c35