March 2022

Securing AI

How Traditional Vulnerability Disclosure Must Adapt

CSET Policy Brief

CSET
CENTER *for* SECURITY *and* EMERGING TECHNOLOGY

AUTHORS
Andrew J. Lohn
Wyatt Hoffman

Executive Summary

When a software developer, government, or independent researcher discovers a cyber vulnerability, they can turn to a broad set of policies, procedures, and informal norms governing responsible disclosure. These practices have evolved over the last 40 years and, in many cases, have helped shorten the period between vulnerability discovery and remediation. As machine learning applications become more widespread, attention has turned to their potential vulnerabilities. While there are relevant analogues to cyber vulnerability discovery and disclosure, ML vulnerabilities are often fundamentally different from their cyber counterparts. It is too early to know whether these ML vulnerabilities will be as impactful as their cyber predecessors, but it is already becoming clear that some of the practices and policies surrounding vulnerability disclosure and management must adapt.

ML models are shaped by data rather than the logic of a programmer, so ML vulnerabilities are often harder to identify precisely, but they can be easy to exploit. Some attacks on ML include altering inputs to cause an otherwise properly performing system to err, tampering with the underlying ML model or its training data, stealing the ML model or training data, and exploiting bugs in the ML software that grant access to the computers running it.[1]

By considering how a range of vulnerability disclosure policies apply to a set of prototypical ML vulnerabilities, we identified four ways that ML vulnerabilities tend to differ from traditional cyber ones:

1. **Patching may be more difficult for ML vulnerabilities**: Patches for ML vulnerabilities often only partially close the security hole while incurring significant expense or performance penalties. Because of these challenges, disclosures should emphasize mitigations over patches.

2. **Many ML systems may be inherently vulnerable**: Because vendors may be unable or unwilling to patch ML systems, they are often inherently vulnerable. When vendors no longer support legacy software, such as older Windows XP systems, there are greater risks that demand more attention from users—the same is true for ML systems.

3. **ML vulnerabilities can be specific to each user**: ML systems are often tailored to each user and adapt over time. Because of this feature, a vulnerability might be unique to one user's model and not apply to other users even if they all used the same base model. Because each model is slightly different, it will be harder to verify whether a base model, or one built from it, is secure.

4. **Proof-of-concept exploits for ML are less practically useful**: ML exploits tend to be brittle and have high failure rates. To date, they have not been as disruptive in the real world as they appear to be in the academic or laboratory setting. They tend to be more useful as warnings to improve security than as viable threats.

These differences lead to several conclusions for vulnerability discoverers, authors of disclosure guidance, and the ML security community writ large:

Findings:

- **Expect slower and imperfect patches and fixes.** The broader security community should adjust expectations away from a simple process that delivers effective fixes with little performance cost.
- **Expect a shift in responsibility for mitigation and remediation toward users and away from vendors.** Vendors' inability to secure their systems means that more security decisions must be made by users or their organizational security teams.

Recommendations:

- **Improve risk assessment for ML applications.** The security decisions that users or their security teams make will involve difficult tradeoffs. They will require a deeper understanding of ML vulnerabilities, the actors who might exploit them, and their impacts.
- **Focus on developing broader mitigation strategies beyond patching.** The shift toward mitigations from patches will require the security community to place more emphasis on developing these defenses and techniques. This reprioritization can better inform users by leading to disclosures that might otherwise go unreported. It also justifies a broader defense-in-depth approach for ML that emphasizes resiliency.
- **Inform and empower users and security managers rather than just vendors when disclosing vulnerabilities.** Presently, vulnerability disclosures are primarily aimed at informing vendors but should focus more on informing users of ML systems.
- **Prioritize proof-of-concept exploits.** For traditional software, defenders can determine the presence or extent of vulnerabilities without exploiting them.

That seems less true for ML systems, which are more likely to be uniquely tailored to their users. ML exploits also currently tend to ignore real world challenges like gaining access to conduct attacks or the effect of varied lighting conditions, noisy data transmission, or redundant sensors. Together, these factors imply that disclosures should make a greater effort to provide exploits for ML vulnerabilities.

Introduction

Who is to blame if a self-driving car is tricked into driving into oncoming traffic or if automated trading algorithms are sent into another flash crash? The perpetrator who triggered the attacks may be the only one to receive a court date, but how responsible is a developer who knew the attack was possible and didn't fix it? What about the researcher who invented the technique or the engineer who uploaded the attack code to the internet? Are governments that know about these vulnerabilities obligated to disclose them? Such questions have been debated since the advent of digital attacks. Although that debate continues, decades of experience have resulted in formal and informal guidelines for assessing whether, when, and how to disclose these dangers to the public. Now, as ML reshapes the software landscape and introduces new types of vulnerabilities, these old questions deserve a new look.[2]

There are many actors in this drama. The initial vulnerability discoverers may be researchers in government or industry, or private hackers who then follow various formal or informal guidelines to decide what to do with that information. They may delay providing some information to various stakeholders, including users, vendors or maintainers. A discoverer's goal in most cases is to reduce the risk of the new vulnerability being used maliciously. As ML vulnerabilities become more prominent, the disclosure guidelines will adjust, and they may be different from those for traditional software because of differences in the vulnerabilities themselves.

Common Vulnerabilities and Remediation

Security professionals have been conditioned over many years toward a set of expectations for vulnerabilities. Traditionally, vulnerabilities arise from programming mistakes. In many of these cases, once the oversight is discovered, a programmer can correct it with a relatively simple and localized change to the code.

Those changes to the code become a "patch" designed to eliminate the vulnerability. Some fixes are part of automatic updates, but often the decision is up to the user or their security team to detect which vulnerabilities are in their systems and prioritize which ones to remediate. Remediation can occur through a number of means, including adopting the patches that exist, implementing other mitigations and workarounds, or, in the most extreme cases, removing the vulnerable software. Although this is the basic mental model that many security professionals rely on, this simple framework does not always apply in reality.

Addressing ML vulnerabilities rarely fits neatly into this relatively straightforward process. For starters, ML models are shaped by data rather than the programmer's code, so there is rarely a simple and localized programming error that can be fixed. There are many possible ML attacks, including altering the inputs to ML systems and causing an otherwise properly performing system to err, stealing the ML model or training data, and exploiting bugs in the ML software that grant access to the computers running it.[3]

We highlight four ways that the processes for addressing ML vulnerabilities might differ from the traditional ones associated with cyber vulnerabilities. These four differences are not an exhaustive list and exceptions exist for each of them. These exceptions—where traditional cyber vulnerabilities are similar to ML ones—provide historical examples of the types of challenges to expect for the future of ML vulnerabilities.

Difference #1: Patching May Be More Difficult for ML Vulnerabilities

ML vulnerabilities will likely rely more on mitigations and imperfect risk-reduction strategies rather than patches that fully resolve a vulnerability. Following too closely to existing vulnerability-handling procedures that prioritize patching could leave users unaware of the risks and slow the development of risk-reduction methods for unpatchable vulnerabilities.

According to the Software Engineering Institute, "In most cases [for traditional software], knowledge of a vulnerability leads the vendor to create and issue a fix to correct it."[4] The cybersecurity world has often struggled when vulnerabilities are disclosed without an effective way to patch them. For example, the Spectre and Meltdown vulnerabilities occurred in hardware rather than software, so the fixes were slow to come and made computers run more slowly when they did.[5] For ML software, fixes often do not exist and the partial solutions that do exist can reduce performance or impose prohibitive costs.

For example, image classifiers are easily tricked by adding subtle noise patterns. There are no complete fixes for these vulnerabilities, and the partial fixes that do exist come at the cost of performance. In one case, the winning entry in a competition to fix this inherent weakness sacrificed accuracy, which fell from 79 percent to 62 percent, and drove up the computing budget due to additional training on over a billion images. Despite the increased cost and decreased performance, the systems were still susceptible to the same types of alterations more than half of the time. These challenges extend beyond image classifiers. More broadly, for a range of ML applications, fixes provide only modest security improvements while creating big performance impacts.[6]

Even if this retraining was effective, it might not be feasible for the immense ML models being fielded. OpenAI, creator of some of the world's largest ML models, noticed a bug affecting its marquee product, GPT-3.[7] Correcting the dataset and retraining the model was an option, but the company decided that the bug was not critical enough to spend the estimated $4.6 million needed for the retraining.[8]

Existing cyber vulnerability disclosure guidance prioritizes patchability. For example, U.S. government policy for discovered vulnerabilities describes "the expectation that [vulnerabilities] will be patched," and the CERT guide for private companies similarly emphasizes the critical role of patches and fixes.[9] Both documents do acknowledge the importance of other mitigations when patches come slowly or not at all, but for ML vulnerabilities, that prioritization must be reversed since patches may not be on the horizon.

Difference #2: Many ML Systems Will Be Inherently Vulnerable

Refocusing on mitigations and risk-reduction rather than patching also weakens a common argument against vulnerability disclosure and re-orients disclosure toward different stakeholders. Many deployed ML systems are inherently vulnerable, in the sense that they contain vulnerabilities that the vendor recognizes but does not or cannot fix. They may go unfixed for several reasons: (1) perfect fixes do not always exist, (2) retraining the models to be more secure can be too expensive to justify, (3) improving security can decrease performance by too much to justify, and (4) securing against one type of attack can increase vulnerability to others.

The first three reasons were discussed in Difference #1. The fourth is still being actively researched, but early indications are that being secure against one type of attack technique can increase vulnerability against another.[10] For example, one defense may protect against attacks that alter images to hide a tank, but that same defense may make it easier for attackers to adjust the images in other ways to make tanks appear where there are none. As a result, even if a patch is developed, the best security decision in some cases might be to ignore it.

In some ways, these unpatched or unpatchable systems are similar to legacy software that is no longer supported by the vendor. In those instances, the user assumes the risk of continuing to use the systems, rather than their upgrades, for either financial or technical reasons. For example, in the 2017 WannaCry attack, the UK's National Health Service was left particularly exposed because of its reliance on older Windows

XP systems for which no patches were available for a known vulnerability.[11] For ML, the unavailability of patches may become even more prevalent.

At least one government, that of the UK, explicitly does not disclose vulnerabilities in software that is unsupported or “vulnerable-by-design.” It claims that “there is no security benefit in fixing a single vulnerability in inherently vulnerable software.”[12] For ML, similar logic could mean an even broader set of systems is exposed with users unaware of the potential risks.

The difficulties in patching vulnerabilities diminish the role of vendors in securing ML systems. Many disclosure processes are currently aimed at informing vendors who may have few options available for securing ML systems, but they are not the only stakeholders who can improve security. The focus of disclosure may need to shift further toward informing users about the degree of risk they are accepting and developing safety and reliability measures for inherently vulnerable systems.

Difference #3: ML Vulnerabilities Can Be Specific to Each User

Many ML systems are tailored to a specific user. Speech recognition systems work best when trained to their user’s voice. An AI-based cyber anomaly detection system can more effectively identify suspicious activity when it is fine-tuned to a user’s normal behavior. This customization means that each user’s system might be slightly different even if they are built from the same base model.

This has two implications for disclosure policies. First, a vulnerability might be unique to one user’s ML model and not apply to other users, even if they all were trained from the same baseline. For instance, if a smart phone’s facial recognition unlock feature adapts to each specific user’s face, one user’s model might be vulnerable to a specific deceptive input that would have no effect on other users’ models.[13] Similarly, a cyber anomaly detection system that adapts to one user’s behavior, system, or network might become vulnerable to an attack exploiting a pattern of behavior unique to that user.

Second, the process of verifying an ML application's security becomes more difficult and complicated because of the uniqueness of individually fine-tuned instances. Traditionally, identical software updates and patches are distributed to all users, and security teams can identify vulnerable users based upon the software versions they have installed. Moreover, when vendors develop software or a patch to it, they will normally publish a corresponding digital fingerprint, also known as a cryptographic hash. The hash allows them to detect any changes to a patch that a malicious actor might make. Used together, the vendors or an organization’s security team can determine which systems are vulnerable and ensure the patch itself is not compromised. But if ML models are unique to each user, it is (1) harder to verify which users are vulnerable, (2) difficult to develop a widely applicable patch, and (3) harder to verify which systems are secure after developing a patch.

Difference #4: Proof-of-Concept Exploits for ML Are Less Practically Useful

The attack code, or proof-of-concept exploit, for conducting a thorough test for an ML vulnerability is not always the highly effective weapon it is made out to be. Even when described as "practical," ML exploits tend to be rather unreliable. Real-world challenges, such as gaining the necessary access to conduct attacks or the means for making any necessary alterations, are assumed rather than demonstrated. And many of the real-world attack scenarios are technically challenging, requiring the attack to succeed over long durations against many different defender configurations, some of which may not be known.

For example, a pattern painted on a mobile missile launcher might try to hide it from an ML identification system, but that deception could fail if just one frame from one wrongly-shaded angle detects the launcher. Proof-of-concept attacks on ML like these typically aspire to show that an attack is not impossible but may not be nearly as effective in real-world settings. Proof-of-concept attacks in traditional digital security typically clear a higher bar by showing that the attack is possible in some favorable conditions.

Conclusions and Recommendations

The differences discussed above have implications for how to handle and disclose ML vulnerabilities, including which ones to prioritize, how to respond to them, and who to empower as part of that response.

Expect slower and imperfect patches and fixes. Traditional computer patches are usually easier to manage than non-patching mitigations or partial fixes. This is because it is relatively easy to identify vulnerable systems, and, once a patch is applied, the system is secured against that vulnerability. ML developers face the unwelcome prospect that future ML vulnerabilities will rely more on the difficult non-patching mitigations for two reasons: First, ML patches may be impossible to develop or require unacceptable tradeoffs such as performance impacts. Second, ML models may be more bespoke in ways that will defy a single patch. Security professionals should adjust their expectations for future ML deployments.

Expect a shift in responsibility for mitigation and remediation toward users and away from vendors. Users and their security teams should expect to play a larger role in securing their own systems, particularly when they have unique instances of particular ML applications. This may require increasing the size and scope of existing security teams at organizations that deploy ML. It will also require greater understanding of generic ML vulnerabilities and how these could affect the specific ML applications under their control. This may also require a greater focus on red teaming and security probes rather than simple scans.

Improve risk assessment for ML applications. These users and security teams are likely to face more difficult decisions that require either trading off performance for security or trading off security of one type for security of another type. Traditional software security teams have to prioritize which fixes to implement but can rely on simply patching the most important ones, often with few downsides. ML systems may require organizations to give more dedicated thought to assessing their risks and deciding which sacrifices to make for security.

Focus on developing broader mitigation strategies beyond patching. Users and organizational security teams will need help developing the best options and configurations to improve security. Much research is centered on trying to develop fixes for ML vulnerabilities but less is being directed toward non-patch mitigations such as preprocessing steps, independent or redundant sensors, or collection procedures for inputs or training data that make tampering more difficult or ineffective. Investment in security research should focus more on alternative mitigations or on developing less-performant but highly secure alternatives.

This shift in focus has implications for the writers of disclosure guidance. Today, some vulnerabilities stay secret when (1) a patch is not possible, (2) when it would not be widely distributed, or (3) when the software already has known but unpatched vulnerabilities. Shifting focus toward mitigations would help justify disclosure in these situations and push defensive efforts in a more practical direction for ML systems.

Inform and empower users and security managers of vulnerabilities rather than just vendors and developers. Prioritizing mitigations and risk assessment over patching places the users and their organizations, rather than the vendors who provide the software, at the center of security. For example, smartphone users are responsible for deciding whether to disable the facial recognition unlocking feature, deciding when to retrain it, and deciding which sensitive data is too risky to have on their phones. Current guidance focuses on how and when to inform vendors. Future guidance should focus more on informing those who are accepting the risk and configuring their systems to balance performance against security.

Prioritize proof-of-concept exploits. For ML applications, security teams will be unable to rely upon traditional network security scans to assess vulnerable systems. Instead, they are more likely to need proof-of-concept exploits to determine how vulnerable their systems are. In response, disclosures will need to focus more on providing proof-of-concept exploits. This is a less risky prospect than it has been for traditional software, because ML exploits tend to be less practically useful.

Authors

Andrew Lohn is a senior fellow with the CyberAI Project at CSET. Wyatt Hoffman co-authored this paper while he was a Research Fellow at CSET. He is currently detailed as an Emerging Technology Policy Fellow in the Office of Emerging Security Challenges at the U.S. Department of State. He completed his contributions to this paper prior to joining the State Department. The views expressed herein are the authors' and do not necessarily reflect those of the U.S. government.

Acknowledgments

For feedback and assistance, we would like to thank John Bansemer, Jim Dempsey, Jonathan Spring, and Gary Corn. Melissa Deng and Alex Friedland provided editorial support.

Document Identifier: doi: 10.51593/2020CA015

Endnotes

[1] Ram Shankar Siva Kumar et al., “Failure Modes in Machine Learning,” *Microsoft*, December 1, 2021, https://docs.microsoft.com/en-us/security/engineering/failure-modes-in-machine-learning; Andrew Lohn, “Hacking AI: A Primer for Policymakers on Machine Learning Cybersecurity” (Center for Security and Emerging Technology, December 2020), https://cset.georgetown.edu/research/hacking-ai/.

[2] James X. Dempsey and Andrew J. Grotto, “Vulnerability Disclosure and Management for AI/ML Systems: A Working Paper with Policy Recommendations” (Stanford Geopolitics, Technology, and Governance Cyber Policy Center, November 10, 2021), https://fsi-live.s3.us-west-1.amazonaws.com/s3fs-public/ai_vuln_disclosure_nov11final-pdf_1.pdf.

[3] Kumar et al., “Failure Modes in Machine Learning”; Lohn, “Hacking AI.”

[4] Allen D. Householder, Garret Wassermann, Art Manion, and Chris King, “The CERT Guide to Coordinated Vulnerability Disclosure” (Carnegie Mellon University Software Engineering Institute, August 2017), https://resources.sei.cmu.edu/asset_files/SpecialReport/2017_003_001_503340.pdf.

[5] Andrew Lohn, “What do Meltdown, Spectre and RyzenFall mean for the future of cybersecurity?,” *TechCrunch*, May 1, 2018, https://techcrunch.com/2018/05/01/what-do-meltdown-spectre-and-ryzenfall-mean-for-the-future-of-cybersecurity/; Rob Wright, “Lessons learned from Meltdown and Spectre disclosure process,” *TechTarget*, August 13, 2018, https://searchsecurity.techtarget.com/news/252446793/Lessons-learned-from-Meltdown-and-Spectre-disclosure-process.

[6] Cihang Xie et al., “Feature Denoising for Improving Adversarial Robustness,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), 501–509.

[7] Tom B. Brown et al, “Language Models are Few-Shot Learners,” arXiv preprint arXiv:2005.14165 (2020), https://arxiv.org/abs/2005.14165.

[8] Chuan Li, “OpenAI’s GPT-3 Language Model: A Technical Overview,” *Lambda*, June 3, 2020, https://lambdalabs.com/blog/demystifying-gpt-3/.

[9] Householder et al., “The CERT Guide to Coordinated Vulnerability Disclosure”; *Vulnerabilities Equities Policy and Process for the United States Government* (Washington, DC: The White House, 2017).

[10] Florian Tramèr and Dan Boneh, “Adversarial Training and Robustness for Multiple Perturbations,” arXiv preprint arXiv:1904.13000 (2019); Florian Tramèr et al., “Fundamental Tradeoffs between Invariance and Sensitivity to Adversarial Perturbations,” arXiv preprint arXiv:2002.04599 (2020).

[11] Alexander Smith, Saphora Smith, Nick Bailey, and Petra Cahill, “Why ‘WannaCry’ Malware Caused Chaos for National Health Service in U.K.,” *NBC News*, May 17, 2017, https://www.nbcnews.com/news/world/why-wannacry-malware-caused-chaos-national-health-service-u-k-n760126.

[12] Government Communications Headquarters, "The Equities Process," November 19, 2018, https://www.gchq.gov.uk/information/equities-process.

[13] Giuseppe Garofalo et al., "Fishy Faces: Crafting Adversarial Images to Poison Face Authentication," *USENIX*, 2018, https://www.usenix.org/system/files/conference/woot18/woot18-paper-garofalo.pdf.