Carnegie Mellon University
Software Engineering Institute

AI HYGIENE STARTS WITH MODELS AND DATA LOADERS

Matthew Churilla, Mahmoud E. Shabana, Renae Metcalf, Shing-Hon Lau

March 2025

Abstract

Cyber hygiene has come to represent the basic tools, processes, and knowledge necessary for operations of a secure and reliable system. Until now, the field of artificial intelligence (AI) has been developing at a fast pace with security a secondary consideration. The absence of a strong emphasis on security has led to the emergence of poor standard practices within the AI community, creating AI systems with inherent vulnerabilities and security issues. This work aims to identify and analyze these problematic practices specifically related to the hygiene of models and data in AI products. Furthermore, it also identifies remedial controls, inspired by traditional cybersecurity principles, to help the AI community strengthen its security posture and improve its overall cyber hygiene.

Introduction

The pace of artificial Intelligence (AI) development has been staggering since the first release of compute unified data architecture (CUDA) in 2007 followed by TensorFlow in 2015 and PyTorch in 2016. These packages have come to form the basis of many machine learning solutions, in both industry and academia, but they were only the first of a series of platforms in the machine learning space. Until now, the AI community has been working hard at solving difficult technical problems and adding features to existing platforms to drive productivity; however, it is time for the community to incorporate security as a vital part of the platforms. Improving AI product security will improve industry security. This poses a question: how can we improve AI security practices from the product level?

With each passing month, the need to establish good security practices and mechanisms in the AI field becomes more obvious. The field of machine learning has repeatedly accepted poor security practices which have led to

- threat models around the use of model zoos[1] such as Tensorflow Hub and PyTorch Model Zoo without signatures or checksums[i]

[1] Model zoos are a collection of trained models that are released for distribution.

- poor practices around deserialization of untrusted models and data[ii,iii]
- the use of models without a log, journal, or manifest detailing on what data they were trained[iv]

This paper focuses on these practices around models and data that machine learning practitioners freely exchange between themselves both within organizations and on the public internet.

AI systems and frameworks are software; therefore, traditional cybersecurity-based vulnerabilities must be appropriately managed. Consequences of cybersecurity vulnerabilities being exploited in AI system include stealing a model file that is at rest or in transit, tampering with data on a filesystem or in a database to change what a model learns, and altering a model file at rest or in transit to influence how a model performs. Google's Secure AI Framework (SAIF)[v] further names and discusses these vulnerabilities in greater detail and describes a secure-by-default framework to be an effective mitigation; however, secure-by-default platforms do not currently exist. Fortunately, there are existing security controls[vi] and procedures[vii] that can help mitigate these traditional cyber risks. Securing models and data loaders, as described throughout in this paper, is the first step on the path to securing a machine learning framework. If implemented, these controls can effectively reduce the risk for both traditional cyber-based and Adversarial Machine Learning[2] attacks and vulnerabilities.

This paper seeks to outline where traditional cybersecurity controls should be available within AI software packages and to identify places where formats and processes should be standardized. By adopting traditional cybersecurity controls, the AI field and community will benefit from decades of research into mechanisms such as encryption, hashing, and checksums, thus reinforcing proven and effective cybersecurity practices as common processes and procedures inherently making AI more secure.

We view models and data loaders as the most pressing and in need of improvement. Models and data loading are the two places in the machine learning workflow where items are frequently loaded from rest or are in transit from local and remote sources thus making them most vulnerable. Items that are at rest or in transit are susceptible to tampering, especially when they are acquired from remote sources of unknown provenance. Cryptographic controls, as well as robust verification mechanisms when working with model files and data, can help secure these parts of the machine learning workflow.

Model files lack integrity, privacy, and authorization mechanisms

A model file is the serialized representation of a machine learning model. Because model files need to exist beyond volatile memory, they must be saved in a format that makes them easy to use later or share with others. When a system is reading in serialized data from possibly unknown sources, there is opportunity for insecure deserialization[viii] issues. The Open Worldwide Application Security Project

[2] Adversarial Machine Learning attacks target the machine learning components in a system. See our previous work on this subject. (reference AML Blog, counter AI document)

(OWASP) is an organization that tracks common vulnerabilities in software applications. Insecure deserialization from untrusted sources has consistently been the OWASP Top 10 lists since 2017[ix]. In response, the field of cybersecurity has established methods, processes, and controls that should be in place when handling deserialization from untrusted sources, such as

- maintaining data integrity via the use of checksums
- preventing deserialization of untrusted or unsafe data or implementation of safe deserialization methods
- using encryption to ensure privacy of serialized sensitive data

Machine learning model files are, at their core, just software. The TensorFlow authors even call this out in their documentation[x]. This means that when you load a machine learning model, you are loading a serialized program. Unfortunately, machine learning platforms will load any model without permission or authorization from the system owner or current user. Our review of the capabilities built into major machine learning platforms shows a lack of mechanisms to enforce authorization policies that specify the conditions under which a model can be loaded and executed. Additionally, there are no built-in mechanisms to verify the provenance of a model file or ensure its integrity through checksum validation against its original source.

In fact, our research has shown machine learning platforms store their models in formats that are susceptible to tampering; models at rest or in transit are open to manipulation by untrusted parties[xi,xii]. Even a model that a party has trained themselves may be unsafe when later deserialized because there is no mechanism in place to help ensure the integrity of the file while it is at rest or in transit.

A model represents a significant amount of research, development, labor, and resources. The parameters contained in the file are often proprietary and sensitive information about operations, customers, or the solution to a difficult problem. Items contained in the model file could leak sensitive information about a model's structure, how it was trained, or its usage. Even if the model is to be made public in the future, an organization may want to keep it private until the intended date of release. Moreover, for models that are completely private, an organization might have regulatory requirements, like the General Data Protection Regulation (GDPR), to further protect the data contained in the model file.

As we look forward, we see more platform providers and more model formats being introduced into the machine learning ecosystem. Until now, each creator has produced their own proprietary formats for model file storage and sharing. Some actors, like Hugging Face and Open Neural Network Exchange (ONNX), are defining standardized formats to make model files more secure and portable; however, even their formats have been shown to have security flaws[xiii,xiv], and the community is not converging on either of these formats but instead continuing to use discrete proprietary formats. With an ever-increasing number of model formats, it will be impossible for consumers to keep up to date with all formats and their respective security risks[xv]. This speaks to the need for standardization across the industry.

So far, we have outlined what we view as some pressing issues for model files:

- Model files are code but are not treated or managed with the same rigor as traditional code.

- Poor practices around serialization and deserialization make them vulnerable to tampering and exploitation.
- The existence of numerous formats, each with distinct and potentially unique weaknesses & vulnerabilities, adds to complexity and risk.

Data loaders do not verify and authorize

As we turn our attention to data and the data centeredness of the machine learning process, we see that machine learning platforms are data centric; however, they do not provide good mechanisms for tracking data provenance nor ensuring data consistency in a standardized way. Without data provenance tracking and consistency verification, it is difficult to determine what training data was used to train a model, whether those data were verified, or whether those data were authorized or approved to be part of the dataset.

Data poisoning[xvi] is a serious adversarial machine learning threat that can create backdoors in machine learning models. The current practice for machine learning systems is to use labels obtained from various annotation file formats as the base mechanism for data loading. These annotation formats do not typically contain checksums for the files that they represent thus allowing an adversary to easily switch or edit data files to suit their purposes. This flaw exists across all public datasets that we know of, regardless of their size. Nicholas Carlini et al. have shown that a lack of data integrity checks means that models can be poisoned even when their training data is web-scale[xvii]. Even though poisoning attacks can be executed through various methods, incorporating robust data integrity checks significantly reduces the attack surface and subsequent risk exposure by limiting the tractability of tampering with models and datasets.

Similar to models, there is no cryptographic signing and chain of custody tracking for annotation or data files, so it is hard to determine their provenance. Without a mechanism to verify the integrity or origin of data, it is not possible to automate authorization, chain of custody, nor integrity checks before the data is used in a machine learning workflow. This means that anyone with access to the annotation files can alter their integrity to add or remove data items into a training process. With the addition of cryptographic signing and change of custody tracking, system owners will be able to determine who authored and authorized data files for use in a machine learning system.

Lastly, there is no automated catalog or journal created by machine learning systems to track the training of a model. This would be especially helpful for consumers of public model zoos where, other than a textual description, there is no record of what data was used to train a model. For example, ImageNet[3] has several versions that have been created over its lifetime, some of which have known weaknesses. Prior to 2020, models trained on ImageNet were found to underperform on

[3] ImageNet is a large image database that has become the primary backbone for computer vision

underrepresented groups of people[xviii]. Because there is no catalog of the data a model has been trained on, there is no way to determine if the ImageNet backbone in a particular model has these issues or not. It is vital that users can easily know on what data a model was trained, validated, and tested.

In this section we have outlined that data loaders should:

- Incorporate checksums and signatures to ensure file integrity.
- Implement minimal chain-of-custody mechanisms that focus on preventing data tampering and at the same time assess its suitability for training.
- Have a data cataloging mechanism that couples data and models together with sufficient clarity to enable a third party to determine what data was used to train, validate, and test a model.

Securing Models and Data Loaders

When trying to implement security for the machine learning process, there is a simple rule of thumb: If you are unsure of the provenance or do not trust the provider of the model or data, then you should not load it, just as you should not load an unknown program on your computer system. Model creators and consumers need a consistent, reliable, and secure way to work with model and data files. To this end, we suggest that whenever a model or data is present, security controls are implemented and utilized in a reliable and verifiable way.

We believe the best place to implement security controls is at the organizational level of the machine learning platform. While engineers of machine learning systems can do this individually, there are issues in doing so. Firstly, it has been shown that correctly implementing security controls is a very difficult task. When implemented by novice personnel, errors can be introduced that increase the amount of risk[xix,xx]. This leaves opportunity for controls to be improperly implemented or circumvented. Secondly, if each engineer or organization discretely implements security controls, this will create an ecosystem of non-compatible implementations in an area where standardization is needed.

Having established that the best place for standardized security controls is in the machine learning platforms themselves, we will present a few controls that should be implemented as a first step to make these platforms more secure by design.

Encryption, such as National Institute of Standards & Technology (NIST) AES-256[xxi], is the gold standard for keeping data private, and it should be implemented in a standardized way on all model formats. This helps producers of models ensure that their models are private while at rest or in transit. Additionally, because the ability to inspect a model is key to performing analysis, encryption helps model producers keep their models more private and secure from adversarial actors such as threats from adversarial machine learning.

Cryptographic checksums and signatures, which are commonly used in cybersecurity, can address two critical issues in securing machine learning model files: verifying the creator of a model file and

detecting any tampering since its creation. We propose that model creation platforms should support a checksum and signing standard to help model consumers understand who created a model and determine whether it has been modified. This also aids Machine Learning Operations (MLOps) to help ensure that model integrity is intact throughout the whole process.

Authorization controls protect systems by denying actions without owners' authorization. Therefore, machine learning platforms should support a standardized model authorization framework based on cryptographic signatures. This helps platform owners create Allow and Deny lists to ensure that only models of their choosing are loaded and executed.

Having looked at how cryptographic methods and authorization mechanism can be used to secure machine learning models, let's now look at how these methods can be used to secure the data loaders that are used to train machine learning systems. We recommend the use of a secure data loader that can be used by consumers to ensure, validate, and enforce robust data management practices.

The first component of a secure data loader is a robust method to verify checksums and confirm the authorization of data items before loading them. Data file checksums should be calculated and verified before reading them into memory for use in machine learning systems. This will ensure that the data used in the system has not been altered from the form meant to be used for training or verification. The data loader must also contain an authorization system that denies loading of files with invalid checksums using signed annotation files and a list of data items that are allowed or denied based on a cryptographic signature. Creators of machine learning systems can use this mechanism to ensure that data provided to the system is allowed and has the intended integrity.

A second component of a secure data loader is a data journaling mechanism that can be used to track what data items a model was trained on, their checksums, and any signatures that are present on the data items. In addition, a journaling mechanism should have the capacity to add its own cryptographic signature to the journal so its integrity can be verified later. This journaling mechanism will help solve issues around data provenance and versioning that exist in modern machine learning pipelines. Ideally, this journal could be included as part of a standardized model file so that future users of the model know the origins of the model.

The last capability we envision in a secure data loader is the ability to handle encrypted data. When trained on public data, there may not be a need for a model and data loader to support loading encrypted data. However, some system owners may want or need to keep data private during the training process. To do this, a secure data loader should support reading encrypted data and properly disposing of it after use.

In this section, we discussed:

- the need for the standardization of model cryptographic capabilities, and we are not alone in this belief; the National Security Agency's (NSA) "Guide for Deploying AI System Securely" also highlights the need for cryptographic methods in the model deployment process[xxii]. For some of these items, the industry is independently moving to create solutions, like ONNX and HuggingFace safetensors, but without standardization, these discrete efforts further fragment the model format landscape.

- the serious and urgent necessity for the industry to come together and standardize on a machine learning model format that provides portability, flexibility, and cryptographic security controls.
- how cryptographic controls and journaling can be built into the data loading process. Without cryptographic capabilities, it is impossible to ensure the confidentiality and integrity of the entire machine learning lifecycle from data ingestion to model creation to deployment.

Conclusions

In this paper we discussed the need to secure machine learning models and the data loading mechanisms used to train them; the real and immediate need for the application of traditional security controls to machine learning platforms; and the need for more standards and collaboration in the machine learning industry. It is vital that these needs be met otherwise the risks to AI systems will continue to grow likely manifesting as security breaches, reputational damage, operational disruptions, and financial loss.

In sum, we offer three suggestions to the machine learning platform providers and major players:

- Provide verified and consistent cryptographic controls, such as encryption, signatures, and checksums, in model and data related processes across all platforms.
- Implement authorization mechanisms to allow system owners to ensure unauthorized content is denied loading into the system.
- Ensure that standardized processes, platforms, and workflows around machine learning products enforce good cyber and AI hygiene practices.

Instead of each organization acting independently to implement these suggestions, we recommend the creation of a community-driven standards body. This body should consist of members from industry, government standards bodies, and academia who will work together to ensure the controls, processes, and procedures that drive AI are consistent, reliable, and secure.

In this paper, we have focused on specific issues where there are viable solutions using traditional cryptographic controls. Additional issues include those found in community interaction with model and data repositories, as well as security concerns found lower in the computing stack, like memory management on GPUs[xxiii,xxiv] and model tampering via machine learning compilers[xxv], that are only recently becoming apparent. These additional security concerns only strengthen the need for a holistic implementation of security mechanisms across the machine learning pipeline. It is our hope that this paper will inform and inspire others to focus on the steps necessary to ensure that AI products and systems are secure by design.

Legal Markings

This material is based upon work funded and supported by the Department of Homeland Security under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center sponsored by the United States Department of Defense.

DM24-1559

Contact Us

Software Engineering Institute
4500 Fifth Avenue, Pittsburgh, PA 15213-2612

Phone: 412.268.5800 | 888.201.4479
Web: www.sei.cmu.edu
Email: info@sei.cmu.edu

[i] Jiang, W., Synovic, N., Sethi, R., Indarapu, A., Hyatt, M., Schorlemmer, T. R., Thiruvathukal, G. K., & Davis, J. C. (2022). *An empirical study of artifacts and security risks in the pre-trained model supply chain*. Proceedings of the 2022 ACM Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses, 105–114. https://doi.org/10.1145/3560835.3564547

[ii] Adia, & Adia. (2024, February 29). Data Scientists Targeted by Malicious Hugging Face ML Models with Silent Backdoor. *JFrog*. https://jfrog.com/blog/data-scientists-targeted-by-malicious-hugging-face-ml-models-with-silent-backdoor/

[iii] Lucas, J. (2024, July 8). *Analyzing the Security of Machine Learning Research Code | NVIDIA Technical blog*. NVIDIA Technical Blog. https://developer.nvidia.com/blog/analyzing-the-security-of-machine-learning-research-code/

[iv] Hardinges, J., Simperl, E., & Shadbolt, N. (2023c). We must fix the lack of transparency around the data used to train foundation models. *Harvard Data Science Review*, *Special Issue 5*. https://doi.org/10.1162/99608f92.a50ec6e6

[v] Google SAIF (n.d.). Risks - SAIF: Secure AI Framework. https://saif.google/secure-ai-framework/risks

[vi] Barker, E., & Barker, E. (2016). *Guideline for using cryptographic standards in the federal government: Cryptographic mechanisms*. US Department of Commerce, National Institute of Standards and Technology.

[vii] Nist, G. M. (2023). *The NIST Cybersecurity Framework 2.0*. Gaithersburg, MD: US Department of Commerce.

[viii] *Insecure deserialization | Web Security Academy*. (n.d.). https://portswigger.net/web-security/deserialization

[ix] OWASP. *A08 Software and Data Integrity Failures - OWASP Top 10:2021*. Owasp.org, 2021, owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/

[x] Tensorflow. (n.d.). *tensorflow/SECURITY.md at master · tensorflow/tensorflow*. GitHub. https://github.com/tensorflow/tensorflow/blob/master/SECURITY.md

[xi] Slaviero, M. *Sour Pickles Shellcoding in Python's Serialization Format*. 2011-07-12)[2012-08-20]. https://media. blackhat. com/bh-us-11/Slaviero/BH_ US 11 Slaviero_Sour_Pickles_Slides. pdf.

[xii] Gabe. (2024, November 14). *Weaponizing ML Models with Ransomware*. HiddenLayer | Security for AI. https://hiddenlayer.com/research/weaponizing-machine-learning-models-with-ransomware/

[xiii] Sestito, K. (2024, November 14). *Hijacking Safetensors conversion on hugging face*. HiddenLayer | Security for AI. https://hiddenlayer.com/research/silent-sabotage/

[xiv] Grigorieva, N. M. (2024, May). *In the Shadow of Artificial Intelligence: Examining Security Challenges, Attack Methodologies, and Vulnerabilities within Machine Learning Implementations*. In 2024 XXVII International Conference on Soft Computing and Measurements (SCM) (pp. 147-150). IEEE.

[xv] Azure. (n.d.). *Abusing ML model file formats to create malware on AI systems: A proof of concept*. GitHub. https://github.com/Azure/counterfit/wiki/Abusing-ML-model-file-formats-to-create-malware-on-AI-systems:-A-proof-of-concept

[xvi] *Papers with Code - Data Poisoning*. (n.d.). https://paperswithcode.com/task/data-poisoning#:~:text=Data%20Poisoning%20is%20an%20adversarial%20attack

[xvii] Carlini, N., Jagielski, M., Choquette-Choo, C. A., Paleka, D., Pearce, W., Anderson, H., ... & Tramèr, F. (2024, May). *Poisoning web-scale training datasets is practical*. In 2024 IEEE Symposium on Security and Privacy (SP) (pp. 407-425). IEEE.

[xviii] *ImageNet*. (n.d.). https://image-net.org/filtering-and-balancing/

[xix] CISA. (2022). *Weak Security Controls and Practices Routinely Exploited for Initial Access*. https://www.cisa.gov/sites/default/files/publications/AA22-137A-Weak_Security_Controls_and_Practices_Routinely_Exploited_for_Initial_Access.pdf

[xx] Woody, C., & Creel, R. (2021). *P21-071: Challenges in building and implementing an effective cybersecurity strategy*. Software Engineering Institute, Carnegie Mellon University, available at: https://apps. dtic. mil/sti/pdfs/AD1126968. pdf.

[xxi] National Institute of Standards and Technology. "*Advanced Encryption Standard (AES)*." CSRC, 9 May 2023, csrc.nist.gov/pubs/fips/197/final.

[xxii] U.S. National Security Agency's Artificial Intelligence Security Center. (2024). *Deploying AI Systems Securely,* https://media.defense.gov/2024/Apr/15/2003439257/-1/-1/0/CSI-DEPLOYING-AI-SYSTEMS-SECURELY.PDF

[xxiii] *CERT/CC Vulnerability Note VU#446598*. (n.d.). https://kb.cert.org/vuls/id/446598

[xxiv] Hoover, J. (2022). *Analysis of GPU Memory Vulnerabilities*.

[xxv] Clifford, E., Shumailov, I., Zhao, Y., Anderson, R., & Mullins, R. (2024, April). ImpNet: Imperceptible and blackbox-undetectable backdoors in compiled neural networks. In *2024 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)* (pp. 344-357). IEEE.