Collecting, Linking, and Assessing Machine Learning Open-Source Software: A Large Scale Collection and Vulnerability Assessment Pipeline

Ben Lazarine
Indiana University
belazar@iu.edu

Srikar Pulipaka
Indiana University
spulipa@iu.edu

Sagar Samtani
Indiana University
ssamtani@iu.edu

Ramesh Venkataraman
Indiana University
venkat@iu.edu

Abstract

In recent years, Artificial Intelligence (AI) has seen rapid advances in performance and impact, disrupting major industries, including finance and healthcare. Machine learning open-source software (MLOSS) platforms such as GitHub and Hugging Face have contributed significantly to this advancement, enabling AI developers to share, reuse, and collaborate on AI development. While these platforms accelerate AI development, the MLOSS assets they host also contain vulnerabilities that can impact applications that leverage them. To map the MLOSS landscape and understand the vulnerabilities contained within MLOSS on platforms such as GitHub and Hugging Face, we have developed an MLOSS Collection Pipeline. Our pipeline has collected 373,634 models from Hugging Face and 39,115 repositories from GitHub and identified 6,751,739 vulnerabilities. The results of our pipeline offer several promising directions for future research, including vulnerability linking analysis and cross-platform vulnerability propagation identification.

Keywords: Artificial Intelligence, Open-source Software, Cybersecurity, AI Risk Management.

1. Introduction

Artificial Intelligence (AI) has staked its ground as a critical and disruptive technology over the last decade. Significant quanities of AI development has moved to open-source platforms such as GitHub and Hugging Face (Gonzalez et al. 2020, Kathikar et al. 2023). These platforms host a range of machine learning open-source software (MLOSS) assets that support AI applications in each stage of their development. GitHub repositories host documentation, data, conventional source code, and AI-specific source code that support the construction of training datasets, model architectures, model source code, and model training and evaluation processes. Hugging Face hosts data, pre-trained model parameters, and model inference endpoints that support the use of powerful pre-trained models (e.g., Llama 3), facilitate the integration of pre-trained models into applications, and allow for such models to be fine-tuned for new uses.

While MLOSS assets significantly accelerate the development of AI, they also bring a unique set of security risks and challenges. Firstly, MLOSS assets are faced with security risks that affect traditional OSS, such as susceptibility to software supply chain attacks (Ladisa et al. 2023), exposing vulnerabilities for public discovery (Go et al. 2023), and malicious contributions directly to repositories and libraries (Goodin 2024). In the context of AI, the potential impact of these security risks is amplified as AI is often deployed in high-impact applications (Bengio et al., 2023). In addition to traditional OSS vulnerabilities, MLOSS faces AI-specific vulnerabilities such as insecure deserialization, insecure model training, and insecure inference endpoints. These vulnerabilities introduce a new set of attack vectors that hackers can exploit to steal sensitive data and manipulate model performance (MITRE 2023). Indeed, the AI Risk Management Framework (RMF) produced by the National Institutes of Standards and Technology (NIST) has clearly indicated the need to effectively govern, map, measure, and manage the security risks associated with AI source code and models. Thus, an infrastructure to collect AI model source code and assess their vulnerabilities is critically needed to facilitate important research that aligns with the principles of NIST's AI RMF.

In this paper, we present our MLOSS Collection Pipeline developed to systematically collect, link, and assess prevailing MLOSS assets to build a comprehensive view of the MLOSS landscape and its associated security risks. Our pipeline has resulted in a collection of 373,634 Hugging Face models, 39,115 associated GitHub repositories, and 6,751,739 vulnerabilities. The pipeline opens promising opportunities for research and practice, providing researchers with the data required for AI security studies, such as linking static vulnerabilities to dynamic vulnerabilities and cross-platform vulnerability propagation. In addition, the results produced from the pipeline offer a significant practical contribution to security teams at organizations looking to integrate AI into their GitHub or Hugging Face by helping them map the risks they may be introducing.

The remainder of this paper is as follows. First, we review MLOSS platforms, prevailing AI vulnerability scanners, and existing MLOSS data collection approached and challenges. Second, we present our MLOSS Collection Pipeline. Third, we present the results of our data collection. Fourth, we discuss several research and practice opportunities that our collection enables, and lastly, we conclude the paper.

2. Machine Learning Open-Source Software Platform Review

2.1. Machine Learning Open-Source Software Platform Overview

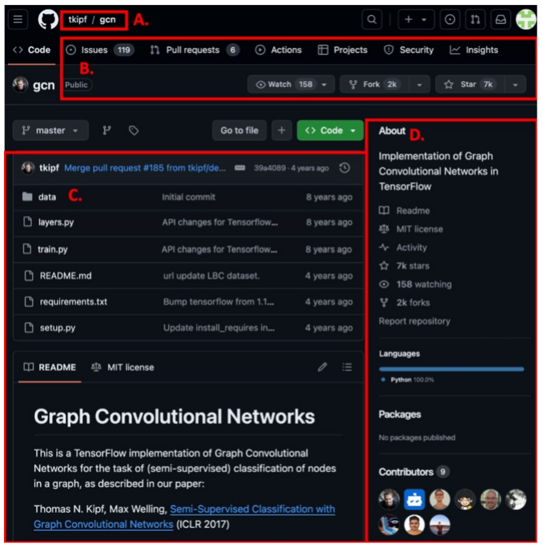
Two major OSS platforms that support MLOSS are social coding repositories (SCRs) and model hubs. Table 1 describes these platforms, the MLOSS components they support, the associated portions of AI application development pipelines they support, and relevant examples.

Table 1. MLOSS Platform Summary

Platform	Description	MLOSS Assets	AI Application	Examples
Social Coding Repositories	Platforms for hosting and sharing documentation, data, and code	Documentation, data, conventional source code, AI-specific source code	Data curation, model training, model validation	GitHub, GitLab
Model Hubs	Platforms for hosting and sharing data sets and ML models	Data, pre-trained model parameters, model inference endpoints	Data curation, model deployment, monitoring	Hugging Face, PyTorch Hub, TensorFlow Hub

Across SCRs and model hubs, AI developers can share, access, and collaborate on a comprehensive set of MLOSS assets needed to support an end-to-end AI application development pipeline, including data curation, model training, model validation, model deployment, and monitoring. SCRs host thousands of AI repositories with MLOSS assets that can be reused, including documentation detailing model construction, training, and usage details, training datasets, conventional source code, and AI-specific source code (e.g., PyTorch (Paszke et al. 2019) and TensorFlow (Dillon et al. 2017). Model hubs host hundreds of thousands of pre-trained ML models that can be used, downloaded, and fine-tuned, in addition to offering training datasets and services to deploy and inference ML models. Each platform is described further in the following subsections.

2.1.1. Social Coding Repositories SCRs offer developers a platform to collaborate and form communities for various application areas (Dabbish et al. 2012). For the remainder of this paper, we focus on GitHub as our primary SCR of interest, as it is a primary SCR leveraged by the AI community (Gonzalez et al. 2020). GitHub offers a rich set of metadata for repositories its hosts detailing their social engagement, contents, descriptions, and key statistics (illustrated in Figure 1 and Table 2).



tkipf Merge pull request #185 from tkipf/de... 39a4089 · 4 years ago

data	Initial commit	8 years ago
layers.py	API changes for Tensorflow...	8 years ago
train.py	API changes for Tensorflow...	8 years ago
README.md	url update LBC dataset.	4 years ago
requirements.txt	Bump tensorflow from 1.1...	4 years ago
setup.py	Update install_requires in...	4 years ago

Implementation of Graph Convolutional Networks in TensorFlow

Graph Convolutional Networks

This is a TensorFlow implementation of Graph Convolutional Networks for the task of (semi-supervised) classification of nodes in a graph, as described in our paper:

Thomas N. Kipf, Max Welling, Semi-Supervised Classification with Graph Convolutional Networks (ICLR 2017)

Figure 1. GitHub Repository Page (A. Identifier; B. Social Engagement; C. Content; D. Description)

All repositories on GitHub have a primary identifier (the name of the repository owner and the repository) and social engagement information at the top of their page. The social engagement metadata of the repository displays critical information about how other GitHub users interact with a repository, including issues they have identified, pull requests they have submitted, and forks they have created. The contents of the repository shown in Figure 1 (tkipf/gcn) highlight the key MLOSS

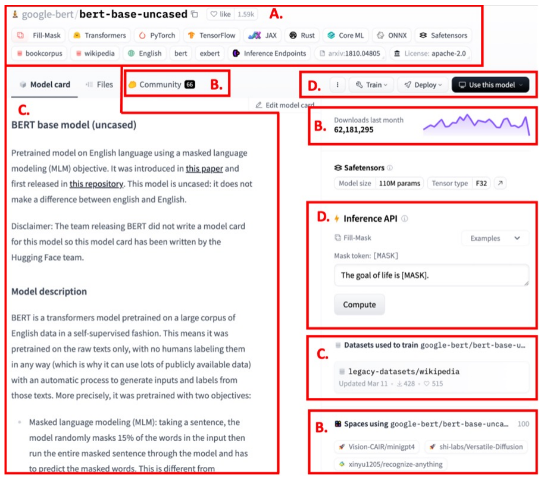
Table 2. Selected GitHub Repository Metadata

Category	Selected Metadata	Description	Data Type
Identifier	Full name	The name of the repository and owner	Text
Social Engagement	Issues	Issues identified by GitHub users	Text
	Pull Requests	Contributions submitted by users	Text, source code
	Forks	How many times the repository has been copied	Discrete
	Watches/Stars	Number of users following the repository	Discrete
Content	Source code	Raw source code posted in the repository	Text, source code
	README/ Documentation	Description of dependencies and how the code operates	Text
	Commit history	Log of changes to the repository	Source code, Date
Descriptive	Description	A brief description of the repository	Text
	Homepage	Link to external site with documentation	Text
	Language	The primary language present in the repository	Categorical
	Contributor	Who has contributed to the repository	Categorical
Statistics	Size	The size of the repository in bytes	Discrete
	Created_at	The date the repository was created	Date
	Updated_at	The date the repository was last updated	Date

components included in AI repositories on GitHub. The source code section shows that the repository shares the data the model was trained on, source code, including the model's architecture (layers.py) and how the model was trained (train.py), and a README that discusses how the model operates. These assets can be used for future AI development.

2.1.2. Model Hubs Model hubs are platforms where AI developers can upload ML models they have trained and use and download models created by others. For the remainder of this paper, we focus on Hugging Face as our primary model hub of interest, as it has been identified as the most popular platform for hosting pre-trained models (Jiang et al. 2023). Similar to GitHub, Hugging Face also has a home page for each model it hosts that displays a rich set of metadata (illustrated in Figure 2 and Table 3).

Identifiers of models on Hugging Face comprise the model's name, the owner's name, and associated tags (e.g., Transformers, Computer Vision, PyTorch, etc.). GitHub repositories and Hugging Face models have some similar content metadata such as READMEs and model cards and commit histories. However, a key difference is that GitHub repositories primarily host source code, while Hugging Face models primarily host model parameters. In An additional major



BERT base model (uncased)

Pretrained model on English language using a masked language modeling (MLM) objective. It was introduced in this paper and first released in this repository. This model is uncased: it does not make a difference between english and English.

Disclaimer: The team releasing BERT did not write a model card for this model so this model card has been written by the Hugging Face team.

Model description

BERT is a transformers model pretrained on a large corpus of English data in a self-supervised fashion. This means it was pretrained on the raw texts only, with no humans labeling them in any way (which is why it can use lots of publicly available data) with an automatic process to generate inputs and labels from those texts. More precisely, it was pretrained with two objectives:

- Masked language modeling (MLM): taking a sentence, the model randomly masks 15% of the words in the input then run the entire masked sentence through the model and has to predict the masked words. This is different from

Figure 2. Hugging Face Model Page (A. Identifiers; B. Social Engagement; C. Content; D. Usage Information)

Table 3. Selected GitHub Repository Metadata

Category	Selected Metadata	Description	Data Type
Identifiers	Full name	The name of the model and owner	Text
	Tags	Denote model type, frameworks used, datasets used, etc.	Text
Social Engagement	Community	Contains discussion between users and pull requests	Text
	Downloads	Model downloads in the last month	Discrete
	Spaces Using	Shows Hugging Face Apps built with the model	Models
Content	Model Card	Model description and usage information	Text
	Files	Model parameters, configuration files, and data	Text, source code
	Storage	Download size	Text
	Commit history	Log of changes to the model	Text, Date
	Datasets Used	Points to datasets used for training that are hosted on Hugging Face	Text
Usage Information	Train	Allows for custom model optimization and fine-tuning	Hyperlink
	Deploy	Enables model deployment on Hugging Face, Amazon, Azure, and Google Cloud	Hyperlink
	Use this model	Shows how to import the model through Transformers	Code
	Inference API	Allows for direct model usage	NA

differentiating feature of Hugging Face is its integrated model usage capabilities. Through a range of options (e.g., Train, Deploy, Use this model, and Inference API), Hugging Face makes it easy for AI developers to use models it hosts, fine-tune them, or deploy them in new applications. It is important to note that while hundreds of thousands of models are hosted on Hugging Face, many do not have all the fields described in Table 3 populated.

2.2. Machine Learning Open-Source Software Vulnerability Assessment Tools

As mentioned above, while the MLOSS assets hosted on GitHub and Hugging Face enable rapid AI development, they also bring a host of vulnerabilities to the AI applications that leverage them. Vulnerability assessment tools such as vulnerability scanners are commonly used to identify and address these vulnerabilities. There are two primary types of AI vulnerability scanners: static and dynamic (Kaur & Nayyar 2020). Static scanners operate by assessing source code directly with rule sets to identify vulnerabilities, including potential secret leakage, insecure permissions and functions, susceptibility to web attacks, and AI-specific vulnerabilities. Dynamic Scanners assess an ML model during runtime, allowing them to identify vulnerabilities that may only manifest during performance, such as susceptibility to text attacks.

Prevailing AI vulnerability scanners that perform static assessments include Bandit, Flawfinder, Semgrep, and ProtectAI. Bandit and Flawfinder scan for traditional source code vulnerabilities in Python and C/C++, respectively. SemGrep scans for both traditional and AI-specific vulnerabilities in all file types, and ProtectAI performs static and dynamic scans for AI-specific vulnerabilities. Dynamic scans are out of the scope of this study as prevailing dynamic scanning tools (e.g., Microsoft Counterfit) are currently not suitable for scanning models at a large scale. However, static scanners typically return vulnerabilities with severity scores that fall into several categories, including low, medium, high, and critical. Such severity scores often are based on CVSS standards and/or agreement from the larger AI security community.

2.3. Machine Learning Open-Source Software Data Collection: Existing Approaches and Challenges

GitHub and Hugging Face both offer generous API access to their data, which has facilitated significant studies on MLOSS (Kathikar et al. 2023, Lazarine et al. 2022). However, with Hugging Face's exponential growth over the last year, there is a need for more comprehensive collections of the models it hosts. Furthermore, with the pace models are added, an incremental crawling approach is necessary to remain current. Beyond Hugging Face, it is also challenging to identify all MLOSS-related repositories on GitHub. Lastly, in addition to collecting GitHub and Hugging Face data, it is also a challenge to assess Hugging Face models for vulnerabilities, as the source code they were developed with is not readily available. These issues motivate the development of a large-scale and automated collection pipeline.

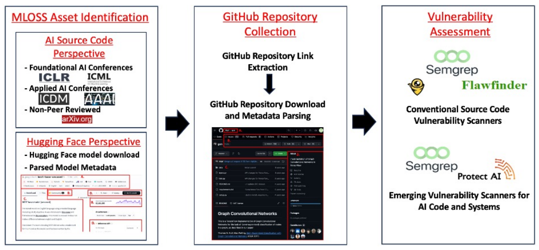
2.4. Machine Learning Open Source Software Vulnerability Assessment: Existing Approaches and Challenges

In addition to past efforts to collect MLOSS assets, recent research has aimed to assess them for potential security issues. Studies have examined AI repositories on GitHub for vulnerabilities, primarily focusing on foundational frameworks (e.g., PyTorch, Tensorflow, etc.). Research on these frameworks has examined how software bugs affect them (Jia et al. 2021), identified commits that may contain CWEs (Harzevili et al. 2022), and mapped the influence of their security issues to other repositories (Sachdeva et al. 2022). However, as previously mentioned, prior vulnerability assessment research has not been done on comprehensive MLOSS datasets. Therefore, there remains a need to identify all MLOSS assets leveraged by AI developers to properly assess the risks posed.

Studies have examined Hugging Face from a security perspective focusing on the platform's adherence to existing OSS best practices, potential security issues with code generation models it hosts, and vulnerabilities in repositories Hugging Face hosts on GitHub. While naming conventions are critical to proper open-sourcing and avoiding attacks such as name spoofing, research has identified that there are significant defects with Hugging Face users' model naming practices (Jiang et al. 2023). Moreover, with open-source large language models (LLMs) hosted on Hugging Face (e.g., Llama 3) being heavily used to generate source code, there have been efforts to assess this code for contained vulnerabilities (Cotroneo et al. 2024). Lastly, past research has examined Hugging Face directly for security issues, performing a vulnerability assessment on the GitHub repositories it hosts on its verified GitHub accound and all of the repositories that have forked them (Kathikar et al. 2023). Similar to vulnerability assessment efforts on MLOSS on GitHub, extant Hugging Face vulnerability assessments have not comprehensively assessed the MLOSS assets that support the development of Hugging Face models.

3. Machine Learning Open-Source Software Collection Pipeline

We present our proposed MLOSS Collection Pipeline in Figure 3. The Pipeline comprises three components: MLOSS Asset Identification, GitHub Repository Collection, and Vulnerability Assessment. MLOSS Asset Identification takes two perspectives: an AI-Source Code Perspective and a Hugging Face Perspective.



MLOSS Asset Identification
AI Source Code Perspective
- Foundational AI Conferences
ICLR
ICML
- Applied AI Conferences
ICDM
AAAI
- Non-Peer Reviewed
arXiv.org
Hugging Face Perspective
- Hugging Face model download
- Parsed Model Metadata
GitHub Repository Collection
GitHub Repository Link Extraction
GitHub Repository Download and Metadata Parsing
Graph Convolutional Networks
Vulnerability Assessment
Semgrep
Flawfinder
Conventional Source Code Vulnerability Scanners
Semgrep
Protect AI
Emerging Vulnerability Scanners for AI Code and Systems

Figure 3. Proposed MLOSS Collection Pipeline

3.1. Machine Learning Open Source Software Asset Identification

3.1.1. AI Source Code Perspective To identify recent AI source code, the first collection perspective of our pipeline identifies top conference venues for foundational and applied AI, as well as non-peer-reviewed public paper sources. The conference venues and sources include the International Conference on Learning Representations (ICLR), Knowledge Discovery in Databases (KDD), Neural Information Processing Systems (NeurIPS), Association for the Advancement of Artificial Intelligence (AAAI), The Conference on Computer Vision and Pattern Recognition (CVPR), The International Conference on Machine Learning (ICML), SIGGRAPH, and Arxiv. This perspective aims to identify and collect the most recent AI repositories on GitHub, as bleeding-edge AI development is published in these conference venues, and many encourage their contributors to share the code associated with their paper (Nature Machine Intelligence 2020). The collection of these papers is automated via scripts that scrape the paper aggregation sites. In addition to identifying AI repositories, we also extract paper metadata, including author names, affiliations, and countries during this collection phase. This data offers interesting insights into key trends in AI research.

3.1.2. Hugging Face Perspective The second perspective of our proposed pipeline uses Hugging Face as a starting point instead of academic AI venues. Hugging Face is another highly active area for AI development, with the platform adding hundreds of thousands of models in the last year. This perspective aims to download the model card of every Hugging Face model and parse the model's associated metadata (Table 3). This collection process is automated via the Hugging Face API and runs incrementally every month to collect new models as they are posted. Similarly to Perspective 1, GitHub repositories are also identified in this process, as many models link to an associated repository in their model card.

3.2. GitHub Repository Collection

Following the identification of MLOSS assets in the first stage of the pipeline, GitHub repository links are extracted from all the papers collected from Perspective 1 and the models collected from Perspective 2. In extracting GitHub links from Hugging Face models, the linkage between the model and the repository is identified. This allows for source code vulnerabilities identified on GitHub to be associated with models on Hugging Face they may impact. Once the GitHub links have been extracted from each set of assets, the associated repositories are cloned, and their metadata are parsed into a database. Cloning each repository is necessary to perform a vulnerability assessment as the vulnerability scanners used in our pipeline operate on local files.

3.3. Vulnerability Assessment

The final phase of our MLOSS Collection Pipeline is a vulnerability assessment of every identified GitHub repository. The vulnerability assessment is conducted with Bandit, Flawfinder, and Semgrep. The three scanners combine to comprise a total of 277 security checks, which can identify 15 vulnerability types across four categories: potential secret leakage, insecure permissions and functions, susceptibility to web attacks, and AI-specific. We summarize the vulnerabilities identified by each scanner in Table 4.

Within each vulnerability category, each scanner also denotes a severity score for each vulnerability, indicating its potential for harm. Further, Bandit returns Common Vulnerabilities and Enumeration (CVE) numbers associated with each vulnerability, Flawfinder returns Common Weakness Enumeration (CWE) numbers, and Semgrep returns both.

4. Data Collection Overview

4.1. Hugging Face

Our MLOSS Collection Pipeline was run iteratively between February 2023 and January 2024. During this time, 373,634 models were collected (over half of the models currently on Hugging Face). Significant model

Table 4. Summary of Vulnerabilities Returned from Scanners

Category	Vuln.	Description	Example	Bandit	Flaw-finder	Sem-grep
Secret	Secret	A potential password/ key	739b6afec22ff801	Yes	No	Yes
	Password	Password found	irods://user:pass	No	No	Yes
	Weak crypto-graphy	Insufficient Crypto. Method	iv_size = crypt.key_size();	Yes	Yes	Yes
	Filetype	File may contain secrets	Django configuration file	No	No	Yes
Insecure Permission and Functions	File per-mission	Dangerous file permissions	int err_code = chmod(filePath, 0664);	Yes	Yes	Yes
	Insecure function	Function can be vulnerable	Use of insecure function (mktemp)	Yes	Yes	Yes
	Insecure module	Module can be vulnerable	Deserialize untrusted data with Pickle	Yes	No	Yes
	Depre-cated library	Library no longer supported	pyCrypto library no longer maintained	Yes	No	Yes
	Insecure connect.	Dangerous internet connections	Call with verify=False disabling SSL certificate checks.	Yes	No	Yes
Web Attack	Insecure input	Dangerous handling of user input	Unsafe yaml load allows arbitrary objects.	Yes	Yes	Yes
	SQL injection	Hardcoded expressions	Unsanitized SQL	Yes	No	Yes
	XML attack	Dangerous XML library	Insecure xmlrpclib use	Yes	No	Yes
	XSS vuln.	Dangerous library usage	By default jinja2 sets autoescape to False.	Yes	No	Yes
AI-Specific	Privacy attack	Sensitive information exposure	Data leaks of sensitive information	No	No	Yes
	Model evasion	Manipulate data	Malicious data modified during security testing	No	No	Yes

metadata and the percentage of models that have each populated are summarized in Table 5.

Each model collected had an associated full name. However, no other field was fully populated, with 76% of the models having some level of content metadata present, 56% denoting the library with which they were developed and very few denoting the languages and datasets with which they were trained (4 and 5% respectively). These results illustrate that, while there are hundreds of thousands of models on Hugging Face, many have sparse details and may be poorly maintained and unlikely to be used.

Table 5. Hugging Face Collection Summary

Metadata	Description	Number of Models	Percent of Models
Full name	Name of the model	373,634	100%
Content	Any content metadata	285,645	76%
Language	Language on which the models are trained on	15,163	4%
Datasets	Dataset on which the model is trained on	18,913	5%
Libraries	Examples: TensorFlow, PyTorch, Transformers	211,078	56%

4.2. GitHub

From Perspective 1, 9,422 papers published between 2021 and 2022 were collected, and 3,393 GitHub repositories were identified and collected. The collected repositories have been forked 415,497 times and have 64,930 open issues, indicating that while less than 4,000 repositories were identified from this approach, the resulting collection supports a very active community. The conferences with the most publications were NeurIPs, AAAI, and CVPR, with 3,325, 1,710, and 1,462 papers, respectively. However, the conference papers with the most GitHub repositories linked were NeurIPs, ICML, and ICLR, with 717, 666, and 643, respectively, and the repositories identified within ICLR papers had the most forking activity.

From the 373,634 Hugging Face models collected in Perspectiv 2e, 241,403 models were identified to be linked to 6,554 unique GitHub Repositories, linking 11% of our Hugging Face collection. Natural language processing (NLP) models, reinforcement learning (RL) models, and audio models had the most links, with 19,782, 10,357, and 3,928, respectively. In addition, 13,339 models that did not have a defined category were also found to have GitHub links.

Additionally, in Perspective 2, we used Hugging Face as a seed to identify additional AI repositories on GitHub. Examining the verified Hugging Face account on GitHub, we identified 111 hosted repositories. These repositories were forked 28,067 times and we added all of these forks to our collection. Lastly, using the GitHub search API with 'huggingface' as a search term, we identified 990 additional repositories.

4.3. Vulnerability Assessment Results

Overall, our vulnerability assessment of the GitHub repositories collected from Perspectives 1 and 2 returned 6,751,739 vulnerabilities. A summary of our vulnerability assessment results is shown in Table 6.

The results of our vulnerability assessment include 624,973 high severity, 530,100 medium severity, and

Table 6. Summary of GitHub Vulnerability Assessment Results

Repository Group	# of Repos	Severity			Total # of Vulnerabilities
		High	Medium	Low	
Perspective 1	3,393	47,842	32,077	244,776	324,695
Perspective 2	6,554	32,640	16,940	460,766	510,346
Hugging Face Root	111	689	1,096	130	1,915
Hugging Face Forks	28,067	537,815	472,304	4,824,765	5,834,884
Hugging Face Searched	990	5,987	7,683	66,229	79,899
Total	39,115	624,973	530,100	5,596,666	6,751,739

5,596,666 low severity vulnerabilities. While the vast majority of the vulnerabilities identified were low severity (83%), across the repositories in Perspective 1 and the Hugging Face forks, there are a significant number of high severity vulnerabilities, with each averaging 14 and 12 high severity vulnerabilities per repository respectively. In comparison, the repositories in Perspective 2, Hugging Face roots, and Hugging Face searched each averaged 5, 6, and 6 high severity vulnerabilities respectively. This may indicate that AI code associated with models published in academic venues (Perspective 1) may be more insecure than AI code written to train models hosted on Hugging Face (Perspective 2). It may also indicate that after forking Hugging Face root repositories, GitHub users either preserve or introduce high severity vulnerabilities.

5. Opportunities for Research and Practice

The results from our MLOSS Collection Pipeline open several promising opportunities for research and practice. Our collection can help AI researchers address this challenge by offering a comprehensive set of static vulnerability assessment results for study. Researchers who perform dynamic assessments can leverage our collection to map (e.g., in manners that align with the AI Risk Management Framework from NIST) static vulnerabilities to dynamic vulnerabilities they may lead to. Successfully performing this mapping can facilitate the prediction of dynamic vulnerabilities from a static vulnerability assessment (which is significantly easier to perform).

A second promising area for study that our collection may facilitate is the identification of cross-platform vulnerability propagation (aligning with the Measure aspect of NIST AI RMF). As AI developers publishing models on Hugging Face leverage source code from GitHub, vulnerabilities may spread across the platforms. Researchers can leverage our collection to identify critically vulnerable repositories on GitHub that are being leveraged to train a large number of the models hosted on Hugging Face, identify how many models GitHub vulnerabilities have spread to, and perform targeted vulnerability remediation. Additionally, developers who engage in insecure development practices across both platforms can be identified and invited to AI security workshops.

Lastly, for practitioners such as security teams within organizations that use AI, our vulnerability assessment can help them improve their AI supply chain security (e.g., aligning with the manage component of the AI RMF). Organizations looking to adopt AI are often faced with the question of whether to incorporate MLOSS. While one drawback to using MLOSS is the security risks associated with OSS practices, the publicly discoverable nature of vulnerabilities can also be leveraged to guide which MLOSS assets may be adopted. Security teams can leverage our collection to ensure that MLOSS assets their organization's leverage are secure or to understand what security risks are being taken on and how to address them.

To facilitate these opportunities for research and practice, a GitHub repository that will contain the source code and instructions needed to instantiate the presented pipeline is forthcoming. Additionally, the repository will provide a template instructing how the pipeline may be launched leveraging cloud resources. Researchers and practitioners will be able to find the repository at https://github.com/benlazarine/mloss-vap.

6. Conclusion and Future Work

With the number of MLOSS assets hosted on platforms such as GitHub and Hugging Face continuing to grow exponentially, this study has presented an MLOSS Collection Pipeline to systematically collect, link, and assess them for vulnerabilities. Our pipeline has resulted in a comprehensive collection of models hosted on Hugging Face, AI repositories on GitHub, and static vulnerabilities they contain. In addition, we discuss promising research and practice opportunities that our resulting collection may facilitate. Future work from this study will continue to incrementally collect Hugging Face and GitHub data with our pipeline. Furthermore, key limitations of our pipeline that may be addressed include incorporating additional vulnerability scanners and identifying additional mechanisms to link Hugging Face models to the source code that was leveraged to train them. For example, incorporating dynamic vulnerability assessments of AI models can potentially return a wider range of results, as they can effectively capture a model's input, output, and task to scan it during runtime. Ultimately, these scan results can help to facilitate additional research opportunities that align with critical national initiatives such as NIST's AI Risk Management Framework.

7. References

Bengio Y., Russell S., and Selman, B. (2023). Pause Giant AI Experiments: An Open Letter. FutureofLife.org.

Cotroneo, D., De Luca, R., & Liguori, P. (2024). Devaic: A tool for security assessment of ai-generated code. arXiv preprint arXiv:2404.07548.

Dabbish, L., Stuart, C., Tsay, J., and Herbsleb, J. (2012, February). Social coding in GitHub: transparency and collaboration in an open software repository. In Proceedings of the ACM 2012 conference on computer supported cooperative work (pp. 1277-1286).

Dillon, J.V., Langmore, I., Tran, D., Brevdo, E., Vasudevan, S., Moore, D., Patton, B., Alemi, A., Hoffman, M. and Saurous, R.A. (2017). Tensorflow distributions. arXiv preprint arXiv:1711.10604.

Go, K. R., Soundarapandian, S., Mitra, A., Vidoni, M., and Ferreyra, N. E. D. (2023). Simple stupid insecure practices and GitHub's code search: A looming threat?. Journal of Systems and Software, 202, 111698.

Gonzalez, D., Zimmermann, T., and Nagappan, N. (2020, June). The state of the ml-universe: 10 years of artificial intelligence & machine learning software development on github. In Proceedings of the 17th International conference on mining software repositories (pp. 431-442).

Gooden, D. (2024). What we know about the xz Utils backdoor that almost infected the world. In Ars Technica.

Harzevili, N. S., Shin, J., Wang, J., Wang, S., & Nagappan, N. (2023, May). Characterizing and understanding software security vulnerabilities in machine learning libraries. In 2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR) (pp. 27-38). IEEE.

Jia, L., Zhong, H., Wang, X., Huang, L., & Lu, X. (2021). The symptoms, causes, and repairs of bugs inside a deep learning library. Journal of Systems and Software, 177, 110935.

Jiang, W., Cheung, C., Thiruvathukal, G. K., & Davis, J. C. (2023). Exploring naming conventions (and defects) of pre-trained deep learning models in hugging face and other model hubs. arXiv preprint arXiv:2310.01642.

Jiang, W., Synovic, N., Hyatt, M., Schorlemmer, T.R., Sethi, R., Lu, Y.H., Thiruvathukal, G.K. and Davis, J.C. (2023, May). An empirical study of pre-trained model reuse in the hugging face deep learning model registry. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) (pp. 2463-2475). IEEE.

Kathikar, A., Nair, A., Lazarine, B., Sachdeva, A., and Samtani, S. (2023, October). Assessing the vulnerabilities of the open-source artificial intelligence (AI) landscape: A large-scale analysis of the Hugging Face platform. In 2023 IEEE International Conference on Intelligence and Security Informatics (ISI) (pp. 1-6). IEEE.

Kaur, A., and Nayyar, R. (2020). A comparative study of static code analysis tools for vulnerability detection in c/c++ and java source code. Procedia Computer Science, 171, 2023-2029.

Ladisa, P., Plate, H., Martinez, M., and Barais, O. (2023, May). Sok: Taxonomy of attacks on open-source software supply chains. In 2023 IEEE Symposium on Security and Privacy (SP) (pp. 1509-1526). IEEE.

Lazarine, B., Zhang, Z., Sachdeva, A., Samtani, S., and Zhu, H. (2022, August). Exploring the Propagation of Vulnerabilities from GitHub Repositories Hosted by Major Technology Organizations. In Proceedings of the 15th Workshop on Cyber Security Experimentation and Test (pp. 145-150).

MITRE. (2023). MITRE ATLAS. https://atlas.mitre.org.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. and Desmaison, A. (2019). Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems, 32.

Research, reuse, repeat. Nature Machine Intelligence 2, 729 (2020).

Sachdeva, A., Lazarine, B., Dama, R., Samtani, S., & Zhu, H. (2022, November). Identifying Patterns of Vulnerability Incidence in Foundational Machine Learning Repositories on GitHub: An Unsupervised Graph Embedding Approach. In 2022 IEEE International Conference on Data Mining Workshops (ICDMW) (pp. 1-8). IEEE.