В тени искусственного интеллекта: изучение проблем безопасности, методологий атак и уязвимостей в реализациях моделей машинного обучения

Н. М. Григорьева

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)

nataliegrigoreva.8@gmail.com

Аннотация. **Искусственный интеллект (ИИ) и модели машинного обучения (МО), несмотря на свою эффективность, подвержены киберугрозам. Модели МО, хотя и представляются как обычные файлы данных, на самом деле являются файлами с кодом, что делает их уязвимыми к атакам. Форматы сериализации, такие как .pickle, .HDF5, .joblib, .ONNX и другие, могут случайно позволять выполнение произвольного кода, что активно используется злоумышленниками. Более того, среды выполнения, такие как PyTorch и TensorFlow, не обеспечивают надежной изоляции, что позволяет создавать вычислительные графы, способные выполнять операции, связанные с обработкой внешних данных, коммуникацией с другими устройствами или программами, а также управлением параллельными процессами. Инструменты разработки программного обеспечения (ПО), такие как ClearML, предназначенные для отслеживания экспериментов и управления версиями моделей, могут добавлять дополнительный уровень риска. Различные версии такого ПО могут иметь уязвимости, что представляет вызов для безопасности систем ИИ и МО. В данной статье исследуются эти проблемы безопасности, возможные атаки, уязвимости и стратегии смягчения рисков для обеспечения безопасности систем ИИ и МО.**

Ключевые слова: искусственный интеллект (ИИ); модели машинного обучения (МО); угрозы безопасности в ИИ/МО; уязвимости ИИ/МО

I. Введение

Искусственный интеллект (ИИ) и машинное обучение (МО) произвели настоящую революцию в различных отраслях благодаря способности обрабатывать огромные объемы данных и принимать решения на основе их анализа и обработки. Однако широкое применение моделей ИИ и МО вызывает опасения относительно их безопасности. Несмотря на то, что эти модели воспринимаются как статические файлы данных, на самом деле они представляют собой исполняемый код, что делает их уязвимыми для различных угроз безопасности. Одной из значительных уязвимостей являются форматы сериализации, обычно используемые для хранения моделей, такие как .pickle, .HDF5, .joblib и .ONNX. Эти форматы, несмотря на удобство для хранения и передачи, могут ненамеренно позволить выполнение произвольного кода. Злоумышленники могут использовать эту уязвимость для выполнения несанкционированного кода в контексте модели, что может привести к возможным утечкам данных или компрометации системы. Более того, среда выполнения моделей, включая такие популярные фреймворки, как PyTorch и TensorFlow, не имеет надежной системы песочницы. Это означает, что модели могут строить вычислительные графы, и выполнять операции ввода-вывода, взаимодействовать с файлами, обмениваться данными через сеть и даже порождать дополнительные процессы. Хотя эта гибкость важна для функциональности моделей, она также открывает пути для потенциальных нарушений безопасности, если не обеспечить их должной защиты. Еще одной потенциальной угрозой является появление инструментов разработки программного обеспечения (SDK), таких как ClearML, разработанных для упрощения разработки и развертывания моделей ИИ и МО. Обе версии, открытая и корпоративная, данной SDK имеют уязвимости, которые, если будут использованы, могут подвергнуть опасности безопасность систем ИИ.

В этой статье исследуются проблемы безопасности, которые могут возникнуть при развертывании ИИ и МО, включая потенциальные атаки, уязвимости и стратегии их устранения. Понимая эти проблемы и внедряя надежные меры безопасности, мы можем обеспечить защиту систем ИИ и МО от потенциальных угроз и обеспечить их безопасное и эффективное развертывание в различных приложениях.

II. Атаки на Hugging Face Safe Tensors.

ИИ и модели МО, хотя и являются мощными средствами ПО, не защищены от угроз безопасности, так как, подобно любой программной системе, могут быть уязвимы к различным векторам атак, которые подрывают их целостность и надежность, и включают в себя т

1. **Адверсальные атаки:** Адверсальные атаки манипулируют входными данными моделей МО для вызова неправильной классификации или некорректного поведения модели. Эти атаки используют чувствительность модели к небольшим, почти незаметным изменениям во входных данных. [1–8]

2. **Отравление данных:** Атаки отравления данных нацелены на искажение обучающего набора данных, используемого для обучения модели МО. Внедряя вредоносные экземпляры данных в обучающий набор, злоумышленники могут манипулировать процессом обучения модели и снижать ее производительность. [9–10]
3. **Кража модели:** Атаки кражи модели направлены на извлечение архитектуры или параметров обученной модели МО. Злоумышленники подают различные входные данные в модель и используют ее выходные ответы для восстановления копии модели, потенциально нарушая права интеллектуальной собственности. [11–12]
4. **Нарушения конфиденциальности:** Модели МО могут непреднамеренно выдать чувствительную информацию о тренировочных данных или отдельных примеров. Нарушения конфиденциальности могут происходить через атаки инверсии модели, когда злоумышленник пытается восстановить чувствительные данные из выходных данных модели. [13]
5. **Уязвимость форматов сериализации модели:** Модели, часто воспринимаемые как обычные файлы данных, на самом деле являются исполняемым кодом, что делает их уязвимыми для атак. Форматы сериализации, такие как .pickle, .HDF5, .joblib, .ONNX и т. д., обычно используемые для хранения моделей, могут случайно позволить выполнение произвольного кода, что активно используется злоумышленниками моделей.

A. Для чего используются Safe Tensors?

Для смягчения упомянутых выше угроз безопасности возникает потребность в надежных и безопасных фреймворках машинного обучения. В то время как проблемы 1–4 хорошо освещены и имеют постоянно развивающиеся решения [14], обращение к уязвимости форматов сериализации моделей (5) является критическим аспектом общей безопасности модели, требующим особого внимания и решений. Эта уязвимость активно используется злоумышленниками, представляя значительную угрозу безопасности и целостности моделей машинного обучения. Атакующий может реализовать функцию эксплойта, воспользовавшись возможностью скрыть ее в небезопасных инструкциях кода модели МО, запакованной в один из небезопасных форматов сериализации. Данная проблема возникает из-за присущего доверия разработчиков и пользователей к безопасности внешних источников данных. Ниже приведены примеры инструкций кода из широко используемых библиотек, которая может привести к случайной загрузке вредоносного кода, позволяющего произвольное выполнение команд [15]:

- *tf.load_library():* Функция load_library() в TensorFlow облегчает загрузку пользовательских операций (ops) из разделяемых библиотек.
- *numpy.load() и numpy.ctypeslib.load_library():* Функция load() в NumPy и метод load_library() в ctypeslib позволяют загружать бинарные данные или разделяемые библиотеки.
- *SessionOptions.register_custom_ops():* В TensorFlow функция register_custom_ops() позволяет регистрировать пользовательские операции.
- *pandas.read_pickle()*: Функция read_pickle() в Pandas облегчает десериализацию данных из файлов pickle.
- *pickle.load() и joblib.load():* И pickle.load(), и joblib.load() отвечают за десериализацию объектов Python из файлов.
- *torch.classes.load_library(), torch.jit.load(), и torch.load():* Эти функции в PyTorch используются для загрузки пользовательских классов, JIT-скомпилированного кода или сохраненных моделей, соответственно.

Упомянутая проблема была частично решена Hugging Face (HF) – платформой и сообществом по машинному обучению и науке о данных, которые помогают пользователям создавать, развертывать и обучать модели ML. Но перед развертыванием модель проходит сериализацию, где она сохраняется на диск в определенном формате. Это бинарное представление позволяет развертывать модель вне исходной среды обучения или дополнительно обучать на новых данных. Эти сериализованные модели, известные как «предварительно обученные модели», значительно способствовали широкому распространению искусственного интеллекта. Они предоставляют командам DS возможность обмениваться, загружать и использовать существующие модели для своих конкретных приложений без необходимости разрабатывать их с нуля. На платформе Hugging Face доступно примерно 500000 моделей. Например, Microsoft и Google имеют около 900 моделей, размещенных там, и, в том числе, работают с ботом для сериализации моделей МО. Часто предварительно обученные модели сохраняются в форматах .pickle, HDF5, joblib, ONNX и т. д., которые, как правило, считаются безопасными при использовании в доверенной среде, например, во внутреннем репозитории команды, но платформы вроде HF к ним не относятся. Одна из причин, по которой эти форматы могут не считаться безопасными, заключается в том, что этот способ сериализации способен выполнять произвольный код Python при десериализации объекта. Это делает его уязвимым для атак, если сериализованные данные поступают из ненадежного источника, поэтому злоумышленник может потенциально создать вредоносный объект .pickle, который выполнит вредоносный код при десериализации. [16–17] Тем не менее несмотря на то, что эти форматы сериализации признаны уязвимыми, они широко используются. Для минимизации возможных рисков безопасности команда HF представила концепцию SafeTensors, [18], которая предлагает многообещающий подход к повышению безопасности моделей МО. SafeTensors предоставляют безопасную альтернативу традиционным тензорам PyTorch [19], предотвращая случайное выполнение вредоносного кода во время обработки входных данных

модели. Использование SafeTensors позволяет обернуть входные данные в безопасный контекст, состоящий только из весов и смещений модели, что блокирует потенциально вредоносные операции, такие как выполнение встроенного кода.

B. Проблемы уязвимости SafeTensors Hugging Face

Интеграция SafeTensors в рабочие процессы машинного обучения позволяет усилить безопасность и устойчивость моделей против различных атак. Однако существуют опасения относительно безопасности конкретного процесса сериализации, используемого SafeTensors. Недавние исследования компании HiddenLayer [20] подчеркивают потенциальные уязвимости в инфраструктуре Hugging Face, демонстрируя, как злоумышленники могут использовать платформу для компрометации моделей, подаваемых на вход службе конвертации. Процесс выглядит следующим образом и показан на рис. 1:

- Атакующий создает собственный репозиторий или загружает существующий с предобученной моделью.
- Атакующий проверяет, есть ли у модели проблемы с сериализацией: если она использует «небезопасные» инструкции кода, упомянутые выше, злоумышленник создает эксплойт и взламывает целевую модель путем внедрения в нее полезной нагрузки.
- Атакующий отправляет отравленную модель в конвертер SafeTensor (SF convertbot) для преобразования модели в «безопасный» формат.
- SF convertbot загружает модель атакующего с использованием метода десериализации torch.load() (рис. 2), сериализует ее, при этом сохраняя внутри модели вредоносную нагрузку, и создает запрос на слияние к владельцу исходного репозитория (легитимного пользователя), предлагая обновить его версию на «более безопасную», которая на самом деле поступает от атакующего!

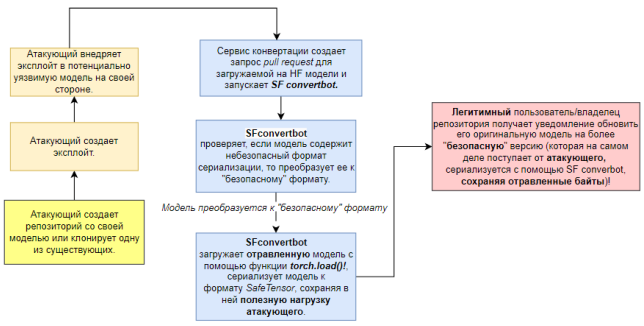

Рис. 1. Определение и понимание вектора атаки на бота сериализации Hugging Face SafeTensor

```
181  def convert_file(
182      pt_filename: str,
183      sf_filename: str,
184      discard_names: List[str],
185  ):
186      loaded = torch.load(pt_filename, map_location="cpu", weights_only=True)
187      if "state_dict" in loaded:
188          loaded = loaded["state_dict"]
```

Рис. 2. Файл Convert.py с небезопасной функцией torch.load() [21]

Файл Converter.py включает метод обнаружения несанкционированных изменений в файлах (Рисунок 3), которые могут указывать на нарушение безопасности или злонамеренную активность, который состоит в проверке размера контрольного файла; если он изменяется более определенного порога (например, 1%), это может вызвать предупреждение о потенциальной проблеме безопасности. Чтобы избежать обнаружения средством контроля безопасности, отслеживающим изменения размеров файлов, злоумышленник, стремящийся внедрить нагрузку в модель, может использовать технику стеганографии [15], чтобы скрыть нагрузку в небольшом количестве байтов. Этот подход позволяет внедрить нагрузку таким образом, чтобы она минимально влияла на общий размер файла, обеспечивая то, что модификация остается в пределах 1 % порога для избежания обнаружения.

```
111  def check_file_size(sf_filename: str, pt_filename: str):
112      sf_size = os.stat(sf_filename).st_size
113      pt_size = os.stat(pt_filename).st_size
114
115      if (sf_size - pt_size) / pt_size > 0.01:
116          raise RuntimeError(
117              f"""The file size different is more than 1%:
118           - {sf_filename}: {sf_size}
119           - {pt_filename}: {pt_size}
120           """
121          )
```

Рис. 3. Функция проверки размера файла с моделью в файле Convert.py [21]

Сценарий, когда атакующий действует в рамках своего репозитория, несет меньшую угрозу, нежели, когда злоумышленник пытается «обновить» репозиторий с моделью легитимного пользователя отравленной моделью. Было продемонстрировано, что злоумышленник может выполнять произвольный код всякий раз, когда кто-то пытается сконвертировать свою модель. Без видимых сигналов для легитимных пользователей их модели могут быть скомпрометированы во время конвертации. Легитимному пользователю официальный бот сервиса конвертации SFconvertbot автоматически отправит запрос на слияние кода из обновленной ветки (pull request). Если пользователь принимает запрос на обновление данных от этого бота, это может привести к замене различных моделей, внедрению нейронных бекдоров, снижению производительности или полному изменению модели. Все это представляет собой значительный риск для цепочки поставок. Кроме того, если пользователь попытается сконвертировать свой частный репозиторий, злоумышленники могут потенциально украсть их токен Hugging Face, скомпрометировать их репозиторий и получить доступ ко всем частным репозиториям, наборам данных и моделям, доступным этому пользователю. Еще один случай, когда платформа HuggingFace бессильна: возможность запуска небезопасного кода в команде, подобной *runpy.run_module*, которая иногда используется для загрузки моделей, подобных .pickle.

III. Уязвимости фреймворков МО

Фреймворки МО, такие как TensorFlow или PyTorch, используют систему исполнения в реальном времени для

выполнения моделей, представленных в виде статических вычислительных графов в TensorFlow или динамических вычислительных графов в PyTorch. Во время исполнения фреймворки машинного обучения выполняют вычислительный граф с использованием предоставленных параметров, от которых зависит его поведение. Сам по себе фреймворк МО не обладает возможностями песочницы, что позволяет ему выполнять различные задачи, такие как операции ввода-вывода, отправка и прием данных по сети и порождение дополнительных процессов в режиме реального времени. Эта гибкость делает фреймворк мощной платформой, но также вносит ряд проблем безопасности. Например, процесс TensorFlow может попытаться выделить больше памяти, чем доступно, что может привести к атаке отказа в обслуживании, если система не подготовлена к таким сценариям. Статья [22] представляет TensorFuzz, фаззер, разработанный для выявления потенциально проблемных входных данных для вычислительных графов TensorFlow, которые могут вызвать ошибки или неопределенное поведение в моделях МО. Методы случайного поиска борются с обнаружением таких данных эффективно, но специализированные инструменты фаззинга, такие как TensorFuzz, могут быстрее выявлять опасные входные примеры. Аналогичные инструменты существуют для моделей PyTorch, такие как Sydr-Fuzz [23], предлагающий непрерывный гибридный фаззинг и динамический анализ. Примеры проблемных входных примеров, приводящих к ошибкам или неопределенному поведению, включают:

- Неправильная классификация изображений (рис. 4)
- Переполнение буфера [24]
- Обход этических политик безопасности в языковых моделях и создание запрещенного или опасного контента [22]

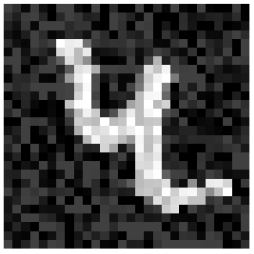

Рис. 4. Изображение, найденное фаззером, которое классифицируется по-разному 32- и 16-битными нейронными сетями. Нижние линии справа способствуют неправильной работе модели

Несмотря на обширные возможности популярных ML-фреймворков для задач машинного обучения, пользователям следует быть внимательными к их потенциальным рискам безопасности и принимать соответствующие меры по их смягчению.

IV. Проблемы безопасности платформ MLOps

Подход MLOps представляет собой набор практик и инструментов, направленных на оптимизацию развертывания, мониторинга и управления моделями машинного обучения в производственных средах. ClearML выделяется как высоко масштабируемая платформа MLOps, известная своей плавной интеграцией с популярными фреймворками и инструментами МО. С помощью ClearML команды специалистов по МО и ИИ управляют процессами разработки моделей в рамках своих проектов, обеспечивая совместную работу, тестирование и итеративные улучшения. Однако ClearML имеет проблемы безопасности, включая скрытые уязвимости нулевого дня в своих открытой и в корпоративной версиях. [25]

Одна из них связана с файлами формата .pickle, подчеркивает риски небезопасной десериализации, описанные выше. [26] Еще одна уязвимость позволяет злоумышленникам перезаписывать файлы на компьютере жертвы во время загрузки файлов. [27] Есть также другие уязвимости, представляющие высокие или критические риски. Самая критическая из них возникает из-за отсутствия аутентификации во всех версиях файлового сервера ClearML. [28] Была также обнаружена уязвимость в хранении пользовательских данных и учетных данных в MongoDB в открытом виде. Несмотря на то, что экземпляры MongoDB обычно недоступны извне, атакующие, получившие доступ к серверу, могут получить данные пользователей ClearML, используя инструмент *mongosh*. [29] Несмотря на успешное устранение всех уязвимостей командой ClearML, критически важно осознавать обширное использование платформ MLOps и необходимость более надежных практик разработки и тщательного тестирования безопасности. Поскольку эти платформы играют все более важную роль в бизнес-операциях, разработчики, ученые по данным и CISO должны понимать потенциальные риски.

V. Заключение

Проведенный анализ выявляет существенные проблемы безопасности в машинном обучении. Уязвимости, подобные тем, что присутствуют в форматах сериализации .pickle, указывают на угрозу выполнения произвольного кода. В то же время, отсутствие надежной изолированной среды во фреймворках PyTorch и TensorFlow позволяет выполнять рискованные операции. Внедрение таких средств, как ClearML, также усложняет и повышает риски МО. Решение этих проблем требует изучения атак, уязвимостей и стратегий смягчения последствий для обеспечения безопасности систем ИИ и МО.

Список литературы

[1] Szegedy, Christian & Zaremba, Wojciech & Sutskever, Ilya & Bruna, Joan & Erhan, Dumitru & Goodfellow, Ian & Fergus, Rob. (2013). Intriguing properties of neural networks.

[2] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Hamessing, ‘Adversarial Examples. arXiv:1412.6572 [stat ML].

[3] Tramèr F, Kurakin A, Papernot N, Goodfellow I, Boneh D, McDaniel P. Ensemble adversarial training: Attacks and defenses. 2018. Paper presented at 6th International Conference on Learning Representations, ICLR 2018, Vancouver, Canada.

[4] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In ICLR, 2017b.

[5] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik and A. Swami, "The Limitations of Deep Learning in Adversarial Settings," 2016 IEEE European Symposium on Security and Privacy (EuroS&P), Saarbruecken, Germany, 2016, pp. 372-387, doi: 10.1109/EuroSP.2016.36.

[6] Wiyatno, Rey & Xu, Anqi. Maximal Jacobian-based Saliency Map Attack. (2018). arXiv preprint arXiv:1808.07945 [stat ML].

[7] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 2017, pp. 39-57, doi: 10.1109/SP.2017.49.

[8] Grigorieva, Natalie M., and Sergei A. Petrenko. "Known Adversarial Attacks by 2023." 2023 Seminar on Information Systems Theory and Practice (ISTP). IEEE, 2023.

[9] Tian, Zhiyi, et al. "A comprehensive survey on poisoning attacks and countermeasures in machine learning." ACM Computing Surveys 55.8 (2022): 1-35.

[10] Yerlikaya, Fahri Anıl, and Şerif Bahtiyar. "Data poisoning attacks against machine learning algorithms." Expert Systems with Applications 208 (2022): 118101.

[11] Oliynyk, Daryna, Rudolf Mayer, and Andreas Rauber. "I know what you trained last summer: A survey on stealing machine learning models and defences." ACM Computing Surveys 55.14s (2023): 1-41.

[12] Wolf, Shaya, et al. "Stealing machine learning parameters via side channel power attacks." 2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). IEEE, 2021.

[13] Dibbo, Sayanton V. "Sok: Model inversion attack landscape: Taxonomy, challenges, and future roadmap." 2023 IEEE 36th Computer Security Foundations Symposium (CSF). IEEE, 2023.

[14] Wang, Y., “Adversarial Attacks and Defenses in Machine Learning-Powered Networks: A Contemporary Survey”, arXiv e-prints, 2023. doi:10.48550/arXiv.2303.06302.

[15] WEAPONIZING ML MODELS WITH RANSOMWARE Available at: https://hiddenlayer.com/research/weaponizing-machine-learning-models-with-ransomware/ (accessed 10 January 2024)

[16] Unsafe Deserialization in Python. Available at: https://knowledge-base.secureflag.com/vulnerabilities/unsafe_deserialization/unsafe_deserialization_python.html (accessed 2 March 2024)

[17] Insecure Deserialization in Python. Available at: https://redfoxsec.com/blog/insecure-deserialization-in-python/ (accessed 10 January 2024)

[18] Huggin Face Safetensors. Available at: https://huggingface.co/safetensors (accessed 7 February 2024)

[19] Introduction to PyTorch Tensors. Available at: https://pytorch.org/tutorials/beginner/introyt/tensors_deeper_tutorial.html (accessed 7 February 2024)

[20] SILENT SABOTAGE. Available at: https://hiddenlayer.com/research/silent-sabotage/ (accessed 23 February 2024)

[21] HuggingFace/safetensors/Convert Available at: https://huggingface.co/spaces/safetensors/convert/blob/main/convert.py (accessed 8 February 2024)

[22] Odena A. et al. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing //International Conference on Machine Learning. – PMLR, 2019. – P. 4901-4911.

[23] Vishnyakov A., Kuts D., Logunova V., Parygina D., Kobrin E., Savidov G., Fedotov A. Sydr-Fuzz: Continuous Hybrid Fuzzing and Dynamic Analysis for Security Development Lifecycle. 2022 Ivannikov ISPRAS Open Conference (ISPRAS), IEEE, 2022, pp. 111-123. DOI: 10.1109/ISPRAS57371.2022.10076861

[24] Eap buffer overflow with torch::load on fuzzy data. Available at: https://github.com/pytorch/pytorch/issues/108865 (accessed 12 March 2024)

[25] MACHINE LEARNING OPERATIONS: WHAT YOU NEED TO KNOW NOW. Available at: https://hiddenlayer.com/research/not-so-clear-how-mlops-solutions-can-muddy-the-waters-of-your-supply-chain/#Identifying-a-Target (accessed 2 March 2024)

[26] NATIONAL VULNERABILITY DATABASE CVE-2024-24590 Detail. Available at: https://nvd.nist.gov/vuln/detail/CVE-2024-24590 (accessed 14 March 2024)

[27] NATIONAL VULNERABILITY DATABASE CVE-2024-24591 Detail. Available at: https://nvd.nist.gov/vuln/detail/CVE-2024-24591 (accessed 14 March 2024)

[28] NATIONAL VULNERABILITY DATABASE CVE-2024-24592 Detail. Available at: https://nvd.nist.gov/vuln/detail/CVE-2024-24592 (accessed 14 March 2024)

[29] NATIONAL VULNERABILITY DATABASE CVE-2024-24595 Detail. Available at: https://nvd.nist.gov/vuln/detail/CVE-2024-24595 (accessed 14 March 2024)