分类号：TP311.5　　单位代码：10335

密　级：无　　学　　号：

浙江大学

博士学位论文

中文论文题目：针对 AI 系统的供应链安全分析与防护

英文论文题目：Security Analysis & Protection for AI System Supply Chains

申请人姓名：

指导教师：

合作导师：

学科 (专业)：网络与信息安全

研究方向：AI 软件与系统安全

所在学院：计算机科学与技术学院

论文递交日期　2026 年 3 月

针对 AI 系统的

供应链安全分析与防护

论文作者签名： _____

指导教师签名： _____

论文评阅人 1：_____

评阅人 2：_____

评阅人 3：_____

评阅人 4：_____

评阅人 5：_____

答辩委员会主席：_____

委员 1：_____

委员 2：_____

委员 3：_____

委员 4：_____

委员 5：_____

答辩日期 _____

Security Analysis & Protection

for AI System Supply Chains

Author's signature： _____

Supervisor's signature： _____

External reviewers: _____

_____

_____

_____

_____

Examining Committee Chairperson:

_____

Examining Committee Members:

_____

_____

_____

_____

_____

Date of oral defence: _____

浙江大学研究生学位论文独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 **浙江大学** 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名: 签字日期： 年 月 日

学位论文版权使用授权书

本学位论文作者完全了解 **浙江大学** 有权保留并向国家有关部门或机构送交本论文的复印件和磁盘，允许论文被查阅和借阅。本人授权 **浙江大学** 可以将学位论文的全部或部分内容编入有关数据库进行检索和传播，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后适用本授权书）

学位论文作者签名: 导师签名:

签字日期： 年 月 日 签字日期： 年 月 日

勘误表

序言

摘要

Abstract

缩略词表

英文缩写	英文全称	中文全称
ZJU	Zhejiang University	浙江大学
ZJU	Zhejiang University	浙江大学
ZJU	Zhejiang University	浙江大学
ZJU	Zhejiang University	浙江大学
ZJU	Zhejiang University	浙江大学
ZJU	Zhejiang University	浙江大学
ZJU	Zhejiang University	浙江大学
ZJU	Zhejiang University	浙江大学

目录

勘误表 I
序言 III
摘要 V
Abstract VII
缩略词表 IX
目录 XI
图目录 XIII
表目录 XV
1 绪论 1
1.1 研究背景及意义 1
1.2 研究现状与目标 4
1.3 本文研究内容与贡献 5
1.4 本文组织与章节安排 5
参考文献 7
附录 9
A 一个附录 9
B 另一个附录 9

图目录

图 1.1　AI 系统架构图..... 1
图 A.1　附录中的图片..... 9

表目录

1 绪论

1.1 研究背景及意义

随着人工智能（Artificial Intelligence, AI）技术的迅猛发展，AI 正在加速融入人类社会的各个领域，并逐渐成为推动社会进步与产业升级的重要引擎。在日常生活中，AI 技术已广泛应用于自动驾驶、智能助手、自然语言处理等关键场景。例如在自动驾驶领域，比亚迪推出的“天神之眼”高阶智能驾驶辅助系统，能够实现全场景的感知与控制辅助功能，为用户提供更加安全、高效的出行体验[1]；在智能助手方面，苹果公司的“Siri 助手”与华为的“小艺助手”能够执行语音指令，完成文件操作、应用启动等任务，显著提升了人机交互的便捷性[2-3]；在自然语言生成领域，OpenAI 于 2022 年发布的 ChatGPT 引发广泛关注，标志着以大参数语言模型（Large Language Model, LLM）为代表的生成式 AI 技术进入高速发展阶段[4]。AI 的广泛应用不仅加速了社会向数字化、信息化与智能化的转型，也成为衡量国家科技竞争力的重要标志。

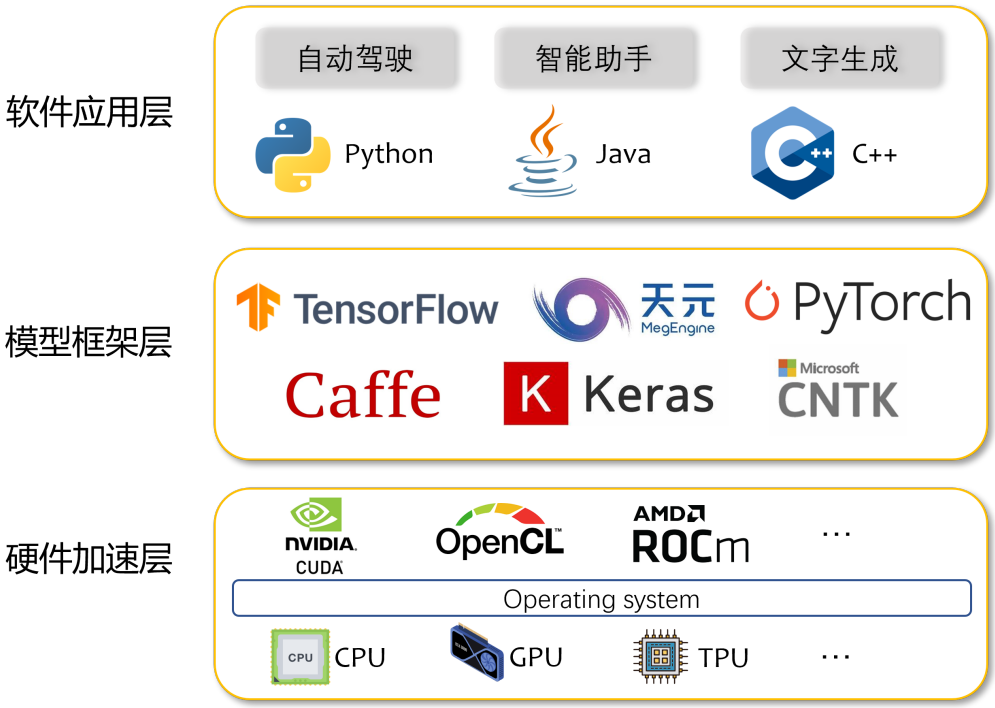

图 1.1 AI 系统架构图

AI 系统的分层架构。随着 AI 技术成体系地持续演化，目前业界研究重点已逐步从单一

模型的性能和结构优化，扩展至模型在真实系统中的集成、部署与运行效率等更为系统性的问题。事实上，在复杂应用环境中，AI 模型往往被嵌入到一个多层次、异构化的 AI 系统中，形成从前端应用到后端算力硬件支持的一体化处理链。所谓 AI 系统，是指由 AI 模型、模型管理软件、运行时环境支持的 AI 框架以及底层硬件资源协同构建而成的综合性技术体系，其核心任务是对图像、语音、文本等输入数据进行智能化分析，并输出相应的决策结果或交互反馈。如图 1.1所示，现代 AI 系统通常由三层组成：软件应用层、模型框架层和硬件加速层，三者之间层层依赖、密切协同，共同构成支撑 AI 服务运行的完整技术栈。

软件应用层处于 AI 系统的最上层，直接面向终端用户，负责构建各类 AI 模型驱动的应用程序。在该层中，开发者主要使用 Python 语言调用预训练的 AI 模型，同时结合 Java、C++ 等高级编程语言实现定制化的业务逻辑和系统功能，例如自动驾驶、人脸识别、智能助手、文字生成等智能服务。这些应用可以通过嵌入式部署或远程服务调用的方式对接模型推理模块，从而灵活适配本地部署或云端服务等不同运行环境。

模型框架层位于 AI 系统的中间层，是连接上层应用与下层硬件的核心支撑组件，承担模型训练、推理与部署的功能。在这一层，开发者通常依赖 TensorFlow、PyTorch 等主流深度学习框架[5-6]，通过其提供的高层 API（多以 Python 形式暴露）定义模型结构，并调用由 C++ 或通用并行计算语言实现的底层算子，高效完成模型计算与参数优化。此外，受限于训练过程对算力资源和高质量标注数据的高昂需求，开发者往往从开源模型平台引入预训练模型，并通过迁移学习或微调的方式实现定制化能力。这一实践在显著提升开发效率和迭代速度的同时，也使模型框架层成为 AI 系统中高度依赖外部资源的关键环节。

硬件加速层位于 AI 系统的最底层，为 AI 模型的算子运算提供实际的运行平台和算力保障。鉴于深度学习模型普遍具有高度并行的计算特性，单纯依赖 CPU 已难以满足性能需求，因此该层通常采用 NVIDIA GPU、Intel NPU、Google TPU 等专用加速硬件。同时，操作系统之上还运行着各类支持通用并行计算的平台，如 CUDA、OpenCL 等。这些平台通过底层驱动与编译器将 AI 框架中的算子编译为 GPU 或 TPU 等硬件指令，并由调度器分配至合适的计算单元，从而实现对模型计算过程的高效加速。

AI 系统的供应链。在 AI 系统中这种多层异构架构显然极大地帮助开发者提升了开发效率和模型运行效率，然而系统的多层级复杂性也引入了高度复杂的供应链关系，使系统

整体暴露于跨层级、跨组件的安全风险之中。在这种分层结构下，每一层均依赖大量第三方库、开源框架或底层驱动组件。当某一层的组件受到攻击或被植入恶意行为时，由于下层为上层提供运行支撑、上层对下层进行功能抽象，这种威胁极易沿着依赖链条向上传播，最终影响整个 AI 系统的安全性与稳定性，造成信息泄露、资产损失甚至服务中断等严重后果。

在软件应用层，开发者为了提升开发效率、减少重复实现，通常会引入大量开源第三方软件包。例如在 Python 生态中，图像处理相关的 AI 应用往往依赖“opencv-python”库[7]，该库提供了丰富且高效的图像处理 API，能够在处理图像时采用高效的算法进行增强、还原、除噪。然而这种对第三方依赖的高度信任也构成了显著的供应链风险，一旦依赖包本身或其间接依赖被恶意投毒，或依赖包包含尚未修复的安全漏洞，恶意代码便可能在模型部署或运行阶段被触发，从而破坏整个 AI 系统的安全边界。

在模型框架层，从头开始训练模型的需要大量显卡算力的硬件支持，以及人工标注的数据集的昂贵成本，因此开发者往往选择基于现有预训练模型进行二次开发，修改模型结构或者对其参数进行微调。这些预训练开源模型广泛来源于 HuggingFace、Model Zoo、TensorFlow Hub 等开源模型平台[8-10]。然而，此类模型来源复杂，且多以二进制格式分发，其内部结构与执行行为对使用者而言往往不可完全验证。一旦模型中被植入恶意后门或隐蔽的可执行逻辑，便可能在推理过程中触发参数篡改、敏感信息泄露，甚至实现任意代码执行，对 AI 系统构成严重威胁。

在硬件加速层中，AI 系统的运行往往使用于不同的加速平台，这些加速平台都依赖底层驱动程序、编译器和固件（firmware）将 AI 算子映射至具体硬件执行逻辑。然而，这些底层组件通常由硬件厂商封闭实现，缺乏透明性，其内部的内存管理机制、计算单元调度方式等细节对用户不可见。一旦这些驱动或固件中存在安全漏洞，或者没有实现特定的安全防护机制，攻击者便可能通过精心构造的模型输入或算子参数触发底层缓冲区溢出，进一步导致权限提升或敏感信息泄露。

综上所述，AI 系统的安全问题已不再局限于单一模型或单一组件，而是深度嵌入于其跨层级、跨组件的复杂供应链之中。因此，构建可信且安全的 AI 系统，必须从供应链全生命周期的角度出发，对各层依赖关系、潜在威胁与防护机制进行系统性分析与设计。

1.2 研究现状与目标

在 AI 系统日益复杂化的背景下，AI 供应链安全问题已逐步受到研究界与工业界的高度关注。

Python 作为 AI 软件开发的主要语言，目前已有大量工作在 Python 软件应用层供应链方面进行了深入研究，它们围绕开源依赖、软件包漏洞与运行时环境的安全隐患。Cheng 等人提出了 PyCRE 框架，采用静态分析方式修复 Python 供应链中存在的错误依赖问题。其核心思路是通过源码分析提取模块之间的依赖关系，并据此判断依赖项的准确性，从而修复依赖冲突和版本不一致等问题，以避免因依赖错误导致的部署失败[11]。Mukherjee 等人则提出了 PyDFix 框架，该方法通过在部署阶段收集运行时信息，实现对依赖冲突的动态检测与修复[12]。此外，Pipreq 作为一个静态依赖生成工具，能够自动从项目源码中推导出所需的依赖列表，用于生成标准化的“requirements.txt”配置文件[13]。在进一步扩展依赖修复范围方面，Ye 等人提出了 PyEGo 框架，其不仅关注软件包本身的依赖问题，还考虑系统环境依赖及 Python 解释器版本兼容性，从而提升整体部署的可复现性与鲁棒性[14]。Cao 等人提出了 PyDC 框架在解决由于 Python 软件中依赖配置错误引起的 Dependency Smell 问题，调查其普遍性、成因以及演化过程。除依赖关系修复外，针对 Python 生态中软件漏洞的分析也是研究重点之一。Mahon 等人提出了 PyPitfall 工具，从整体视角系统分析了 PyPI 生态的依赖结构与漏洞传播关系，揭示了直接依赖与传递依赖对系统漏洞暴露风险的显著影响[15]。Alfadel 等人通过对 698 个 Python 包的 1396 条漏洞报告进行实证分析，发现 Python 软件包的漏洞数量呈上升趋势，且部分漏洞在被发现前的生命周期超过三年[16]。Bogaerts 等人则构建了包含 1026 个已公开 Python 漏洞的数据库，并提取了对应的补丁与易受攻击代码，为后续漏洞检测与修复研究提供数据基础[17]。综上所述，现有研究在软件应用层已提出诸多有效工具和框架，用于修复依赖配置错误、缓解漏洞风险、提升软件包部署的稳定性。然而，这些工作大多聚焦于“已知漏洞”或“显式依赖关系”的分析，并且都是以包为分析粒度，对更细粒度的模块粒度少有分析，且尚未深入探讨供应链机制本身如何被恶意利用的问题。

在模型框架层，研究者逐渐关注到模型本体及其运行框架在 AI 供应链中的关键地位。相较于传统软件包，这一层的攻击面更贴近模型执行逻辑，具备更强的隐蔽性与破坏力。已有多项工作围绕模型文件的代码注入、恶意模型检测、模型转换过程的安全隐

患等方面展开系统研究。

在硬件加速层，AI 系统高度依赖 GPU 等专用硬件资源以实现高性能计算，而 GPU 驱动、固件与用于计算的 CUDA、OpenCL 程序成为新的攻击入口。近年来，随着 CUDA 等平台复杂度的提升，越来越多研究开始探讨底层硬件与驱动对 AI 系统的供应链风险，包括驱动漏洞、微架构攻击、CUDA 程序漏洞检测等。

1.3 本文研究内容与贡献

1.4 本文组织与章节安排

参考文献

[1] 比亚迪. 比亚迪获全国首张 L3 自动驾驶高快速路测试牌照，全面加速智能化布局[EB/OL]. 2023. https://www.byd.com/cn/news/2023/detail496.

[2] Apple Inc. Siri[EB/OL]. 2025. https://www.apple.com/siri/.

[3] Huawei. 小艺助手[EB/OL]. 2025. https://xiaoyi.huawei.com/chat/.

[4] OpenAI. ChatGPT[EB/OL]. 2022. https://openai.com/zh-Hans-CN/index/chatgpt/.

[5] ABADI M, BARHAM P, CHEN J, et al. {TensorFlow}: a system for {Large-Scale} machine learning [C]//12th USENIX symposium on operating systems design and implementation (OSDI 16). 2016: 265-283.

[6] PASZKE A, GROSS S, MASSA F, et al. Pytorch: An imperative style, high-performance deep learning library[J]. Advances in neural information processing systems, 2019, 32.

[7] BRADSKI G, the OpenCV team. opencv-python: OpenCV library for Python[EB/OL]. 2025. https://pypi.org/project/opencv-python/.

[8] INC. H F. Hugging Face Hub: A Platform for Sharing Machine Learning Models, Datasets and Demos [EB]. 2026.

[9] Various Contributors. Model Zoo: A Collection of Pre-trained Deep Learning Models[EB]. 2026.

[10] Google Research. TensorFlow Hub: A Repository of Trained Machine Learning Models[EB]. 2026.

[11] CHENG W, ZHU X, HU W. Conflict-Aware Inference of Python Compatible Runtime Environments with Domain Knowledge Graph[C/OL]//ICSE '22: Proceedings of the 44th International Conference on Software Engineering. Pittsburgh, Pennsylvania: Association for Computing Machinery, 2022: 451-461. https://doi.org/10.1145/3510003.3510078. DOI: 10.1145/3510003.3510078.

[12] MUKHERJEE S, ALMANZA A, RUBIO-GONZÁLEZ C. Fixing dependency errors for Python build reproducibility[C]//Proceedings of the 30th ACM SIGSOFT international symposium on software testing and analysis. 2021: 439-451.

[13] SMITH J. pipreq[EB/OL]. 2023. https://github.com/bndr/pipreqs/.

[14] YE H, CHEN W, DOU W, et al. Knowledge-based environment dependency inference for Python programs[C]//Proceedings of the 44th International Conference on Software Engineering. 2022: 1245-1256.

[15] MAHON J, HOU C, YAO Z. PyPitfall: Dependency Chaos and Software Supply Chain Vulnerabilities in Python[J]. arXiv preprint arXiv:2507.18075, 2025.

[16] ALFADEL M, COSTA D E, SHIHAB E. Empirical analysis of security vulnerabilities in Python packages[J/OL]. Empirical Softw. Engg., 2023, 28(3). https://doi.org/10.1007/s10664-022-10278-4. DOI: 10.1007/s10664-022-10278-4.

[17] BOGAERTS F C G, IVAKI N, FONSECA J. A Taxonomy for Python Vulnerabilities[J/OL]. IEEE Open Journal of the Computer Society, 2024, 5: 368-379. DOI: 10.1109/OJCS.2024.3422686.

附录

A 一个附录

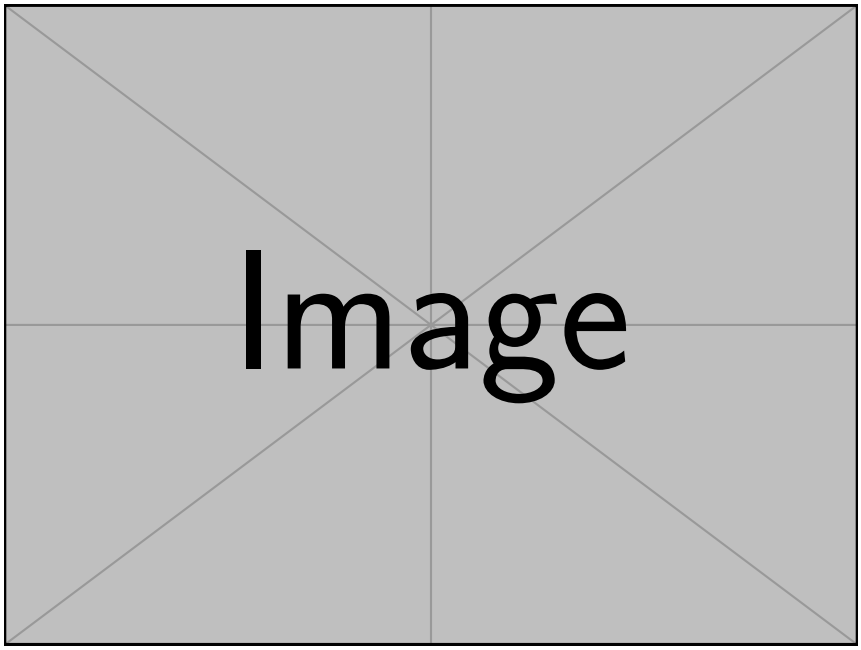

图 A.1 附录中的图片

B 另一个附录