

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 .....	7
1.4 本文组织与章节安排 .....	7
参考文献 .....	9
附录 .....	11
A 一个附录 .....	11
B 另一个附录.....	11



## 图目录

图 1.1 AI 系统架构图.....	1
图 A.1 附录中的图片.....	11



## 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<sup>[1]</sup>; 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<sup>[2-3]</sup>; 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<sup>[4]</sup>。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。

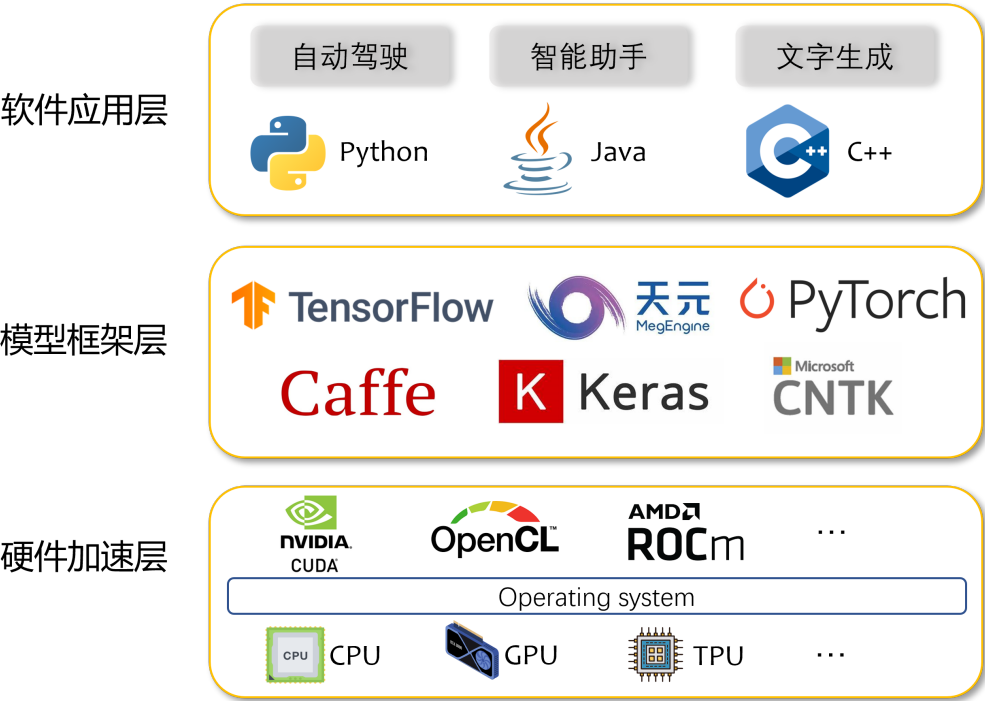


图 1.1 AI 系统架构图

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<sup>[5-6]</sup>,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<sup>[7]</sup>，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<sup>[8-10]</sup>。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。随着 AI 应用从单一模型扩展为由多层组件协同构成的复杂系统, 其安全性也愈发依赖于不同层级组件之间的依赖关系及每一层独有的供应链机制。

**AI 系统软件应用层供应链研究现状。**Python 作为 AI 软件开发中最为主流的编程语言之一, 围绕其软件应用层的供应链安全问题, 也层出不穷, 根据 Sonatype 自 2019 年以来的多年年度报告, 不仅开源软件包的数量在逐年激增, 恶意软件包的数量也随之层出不穷, 截至 2024 年, Sonatype 组织已经发现超过 704,102 个恶意的开源软件包<sup>[11]</sup>。同时报告还指出 CVE 数量也持续呈指数级增长, 开发者却无法跟上这样爆炸级的漏洞增长数, 无法保证漏洞能够被及时修复。有相关研究表明, 部分漏洞在开源软件包中存在的时间甚至长达 3 年以上未修复<sup>[12]</sup>。高风险的开源软件包不仅会对 AI 软件的开发造成影响, 甚至能对整个 AI 系统造成威胁。

目前已有大量研究从开源依赖管理、软件包漏洞以及运行时环境风险等方面展开深入分析。Cheng 等人提出了 PyCRE 框架, 采用静态分析方法修复 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与抽象语法树技术 (Abstract Resource Tree, AST) 提取模块之间的依赖关系, 并结合软件包配置文件构建依赖图, 从而判断依赖图中的依赖项是否存在缺失和冲突, 进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 AI 软件部署失败<sup>[13]</sup>。Mukherjee 等人提出了 PyDFix 框架, 该框架通过在部署阶段收集运行时的控制台信息, 判断安装过程中具体是哪些软件包出现错误, 以及错误类型是依赖缺失还是版本不一致, 并基于这些错误信息实现对依赖冲突的动态检测与修复<sup>[14]</sup>。此外, Pipreq 作为一种静态依赖生成工具, 它可以通过自动化地分析 Python 项目中 `import` 语句引入了哪些模块, 再通过一个一对一的模块与软件包名的映射, 来判断该项目需要哪些软件包, 从而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<sup>[15]</sup>。在进一步扩展依赖修复范围方面, Ye 等人提出了 PyEGo 框架, 该框架不仅关注软件包层面的依赖问题, 还同时考虑系统环境依赖以及 Python 解释器版本兼容性, 从而提升整体部署过程的可复现性与鲁棒性<sup>[16]</sup>。此外, Cao 等人提出了 PyDC 框架, 针对由于 Python 软件依赖配置错误引发的 Dependency Smell 问题展开研究, 系统分析了此类问题的普遍性、成因及其演化过程。

除依赖关系修复外，针对 Python 生态中软件漏洞的分析同样是软件应用层供应链研究的重要方向之一。由于 AI 软件通常依赖大量的核心 AI 组件包和其他开源软件包，这些关键依赖项中潜在的漏洞也是影响 AI 系统安全性的重要因素之一。Mahon 等人提出了 PyPitfall 工具，从整体视角系统分析了 PyPI 生态中的依赖结构及漏洞传播关系，揭示了直接依赖与传递依赖在系统漏洞暴露风险中的显著影响<sup>[17]</sup>。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<sup>[18]</sup>。在更宏观的层面，Ladisa 等人对开源软件供应链的攻击实现了一个系统的分类，该分类独立于特定的编程语言或生态系统，并覆盖了从代码贡献到软件包分发的所有供应链阶段。其以攻击树的形式刻画了 107 种不同的攻击向量，并将其与 94 起真实世界事件及 33 类缓解措施进行映射<sup>[19]</sup>。类似地，Bogaerts 等人则更专注于 Python 语言，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<sup>[20]</sup>。

综上所述，现有研究在 AI 软件应用层已提出诸多有效工具和框架，可以用于自动修复 AI 项目中常见的依赖配置错误、漏洞风险检测、软件包部署的错误等问题，从而提升 AI 软件包的稳定性和安全性。然而，现有工作大多聚焦于已知漏洞或显式依赖关系分析，并且通常都是以软件包为分析粒度，对更细粒度的模块级行为关注度较少，同时也尚未深入探讨供应链机制本身如何被恶意利用的问题。

**AI 系统模型框架层供应链研究现状。**在模型框架层，研究者逐渐关注到预训练模型的本身及其运行框架提供的算子在 AI 供应链中的关键地位。近年来，开源模型库中的模型数量呈爆炸式增长。以 Hugging Face 为例，仅在 2022 年至 2025 年期间，该平台上已累计发布超过 200 万个开源模型<sup>[21]</sup>。如此庞大的模型规模在显著降低模型获取与复用成本的同时，也为恶意模型的传播提供了现实土壤。已有公开报告表明，开源模型库已成为恶意行为者新的攻击载体。据 JFrog 于 2024 年 2 月发布的分析报告，他们在 Hugging Face 平台上发现了超过 100 个恶意模型，涉及 TensorFlow 与 PyTorch 等多个主流深度学习框架。这些模型在加载或推理过程中被用于实现反向 shell、任意文件读写、启动特定程序以及代码执行等恶意行为<sup>[22]</sup>。相较于传统的软件包投毒，模型与框架层面的攻击更贴近模型的实际执行逻辑，往往具有更强的隐蔽性与潜在破坏力。围绕模型框架层的安全风险，已有研究从不同角度展开了系统性探索，主要集中在恶意模型行为分

析、模型安全检测以及框架层漏洞挖掘等方向。

现有工作表明,模型层面的恶意逻辑注入大致可以归纳为两类:一类是传统机器学习语境下的恶意模型,其目标在于操纵模型的推理结果,例如是将智能驾驶模型的推理结果由红灯变为绿灯,从而间接导致交通事故,这类工作主要关注模型预测行为本身的安全性,而非模型执行过程的系统级影响;另一类则是将 AI 模型本身作为恶意载体,利用模型的序列化格式、加载流程或推理执行过程嵌入并触发恶意软件或恶意代码,例如利用模型窃取用户本地文件,读取敏感信息等。传统机器学习语境下,恶意模型攻击主要包括后门攻击,即在训练阶段向模型中植入隐蔽触发器(后门),使模型在正常输入下表现正常,但在触发条件出现时输出攻击者预期的结果<sup>[23-25]</sup>;对抗样本攻击,即通过对输入样本施加微小扰动,从而诱导模型产生错误分类结果<sup>[26-28]</sup>;近年来,随着 LLM 大参数高效微调技术的发展,研究者进一步发现了可以利用 LoRA 等轻量化微调机制注入恶意触发逻辑,从而在不显著改变模型整体性能的前提下实现隐蔽攻击<sup>[29-30]</sup>。

在 AI 模型作为恶意逻辑载体的语境下,攻击者嵌入恶意逻辑的方式主要有三类:一类是将恶意软件或者逻辑嵌入模型的二进制参数或者特定模型的层次结构中,并在模型运行阶段将恶意软件重组和触发运行。Hua 等人提出的 Malmodel 技术<sup>[31]</sup>,将恶意模型嵌入 TensorFlow Lite 模型的层数、覆盖率等参数中,并利用 Java 反射机制主动触发。Hitaj 等人提出的 MaleficNet<sup>[32]</sup>,利用扩频信道编码结合纠错技术,将恶意负载注入深度学习网络参数中。类似地,其他工作如 Evilmodel 1.0<sup>[33]</sup>、Evilmodel 2.0<sup>[34]</sup>以及 StegoNet<sup>[35]</sup>,则采用最低有效位 (Least Significant Bit, LSB) 隐写术来隐藏恶意软件。第二类是将模型嵌入模型的 lambda 层中,这类攻击仅适用于 TensorFlow 这种支持 lambda 层的模型框架,并且极容易被检测出来,因为仅需检测模型是否存在 lambda 层以及判断 lambda 层逻辑是否是恶意的即可<sup>[36-37]</sup>。第三类也是目前攻击者使用最广泛的一类,即使用 pickle 等不安全的序列化格式,将恶意逻辑嵌入二进制模型中,并利用模型反序列化过程中会运行代码的机制来运行恶意逻辑<sup>[38-40]</sup>。

**AI 系统硬件加速层供应链研究现状。**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] Sonatype. State of the 2021 Software Supply Chain[J/OL]. Sonatype Blog, 2021. <https://www.sonatype.com/blog/software-supply-chain-2021>.
- [12] AKHOUNDALI J, NOURI S R, RIETVELD K, et al. MoreFixes: A large-scale dataset of CVE fix commits mined through enhanced repository discovery[C]//Proceedings of the 20th International Conference on Predictive Models and Data Analytics in Software Engineering. 2024: 42-51.
- [13] 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.
- [14] 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.
- [15] SMITH J. pipreq[EB/OL]. 2023. <https://github.com/bndr/pipreqs/>.
- [16] 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.
- [17] MAHON J, HOU C, YAO Z. PyPitfall: Dependency Chaos and Software Supply Chain Vulnerabilities in Python[J]. arXiv preprint arXiv:2507.18075, 2025.
- [18] 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.
- [19] LADISA P, PLATE H, MARTINEZ M, et al. SoK: Taxonomy of Attacks on Open-Source Software Supply Chains[C]//2023 IEEE Symposium on Security and Privacy (SP). 2023: 1509-1526. DOI: 10.1109/SP46215.2023.10179304.
- [20] BOGAERTS F C G, IVAKI N, FONSECA J. A Taxonomy for Python Vulnerabilities[J]. IEEE Open Journal of the Computer Society, 2024, 5: 368-379. DOI: 10.1109/OJCS.2024.3422686.
- [21] RÍOS F. Hugging Face's two million models and counting[EB/OL]. AI World. 2025. <https://aiworld.eu/stories/hugging-face-two-million-models>.
- [22] JFrog Security Research Team. Examining Malicious Hugging Face ML Models with Silent Backdoor [EB/OL]. JFrog Security Research. 2025. <https://research.jfrog.com/examining-malicious-hugging-face-ml-models-with-silent-backdoor/>.
- [23] JI Y, ZHANG X, WANG T. Backdoor attacks against learning systems[C]//2017 IEEE Conference on Communications and Network Security (CNS). 2017: 1-9. DOI: 10.1109/CNS.2017.8228656.

- [24] GU T, LIU K, DOLAN-GAVITT B, et al. Badnets: Evaluating backdooring attacks on deep neural networks[J]. Ieee Access, 2019, 7: 47230-47244.
- [25] TURNER A, TSIPRAS D, MADRY A. Label-consistent backdoor attacks[J]. arXiv preprint arXiv:1912.02771, 2019.
- [26] KURAKIN A, GOODFELLOW I, BENGIO S. Adversarial machine learning at scale[J]. arXiv preprint arXiv:1611.01236, 2016.
- [27] HUANG S, PAPERNOT N, GOODFELLOW I, et al. Adversarial attacks on neural network policies [J]. arXiv preprint arXiv:1702.02284, 2017.
- [28] MADRY A, MAKELOV A, SCHMIDT L, et al. Towards deep learning models resistant to adversarial attacks[J]. arXiv preprint arXiv:1706.06083, 2017.
- [29] YIN M, ZHANG J, SUN J, et al. LoBAM: LoRA-Based Backdoor Attack on Model Merging[J]. arXiv preprint arXiv:2411.16746, 2024.
- [30] LIU H, LIU Z, TANG R, et al. Lora-as-an-attack! piercing llm safety under the share-and-play scenario [J]. arXiv e-prints, 2024: arXiv-2403.
- [31] HUA J, WANG K, WANG M, et al. MalModel: Hiding Malicious Payload in Mobile Deep Learning Models with Black-box Backdoor Attack[J]. arXiv preprint arXiv:2401.02659, 2024.
- [32] HITAJ D, PAGNOTTA G, DE GASPARI F, et al. Do You Trust Your Model? Emerging Malware Threats in the Deep Learning Ecosystem[J]. arXiv preprint arXiv:2403.03593, 2024.
- [33] WANG Z, LIU C, CUI X. Evilmodel: hiding malware inside of neural network models[C]//2021 IEEE Symposium on Computers and Communications (ISCC). 2021: 1-7.
- [34] WANG Z, LIU C, CUI X, et al. Evilmodel 2.0: bringing neural network models into malware attacks [J]. Computers & Security, 2022, 120: 102807.
- [35] LIU T, LIU Z, LIU Q, et al. StegoNet: Turn Deep Neural Network into a Stegomalware[C/OL]// ACSAC '20: Proceedings of the 36th Annual Computer Security Applications Conference. <conf-loc>, <city>Austin</city>, <country>USA</country>, </conf-loc>: Association for Computing Machinery, 2020: 928-938. <https://doi.org/10.1145/3427228.3427268>. DOI: 10.1145/3427228.3427268.
- [36] Splinter0. Tensorflow Remote Code Execution with Malicious Model[EB/OL]. GitHub. 2024. <https://github.com/Splinter0/tensorflow-rce>.
- [37] CERT Coordination Center. Vulnerability Note VU#253266[EB/OL]. CERT/CC. 2025. <https://kb.cert.org/vuls/id/253266>.
- [38] The Hacker News. New Attack Technique 'Sleepy Pickle' Targets Machine Learning Models [EB/OL]. The Hacker News. 2024. <https://thehackernews.com/2024/06/new-attack-technique-sleepy-pickle.html>.
- [39] Pjcampbe11. Pickle-File-Attacks[EB/OL]. GitHub. 2024. <https://github.com/pjcampbe11/Pickle-File-Attacks>.
- [40] Trail of Bits. Exploiting ML Models with Pickle File Attacks (Part 1)[EB/OL]. Trail of Bits. 2024. <https://blog.trailofbits.com/2024/06/11/exploiting-ml-models-with-pickle-file-attacks-part-1/>.

## 附录

### A 一个附录

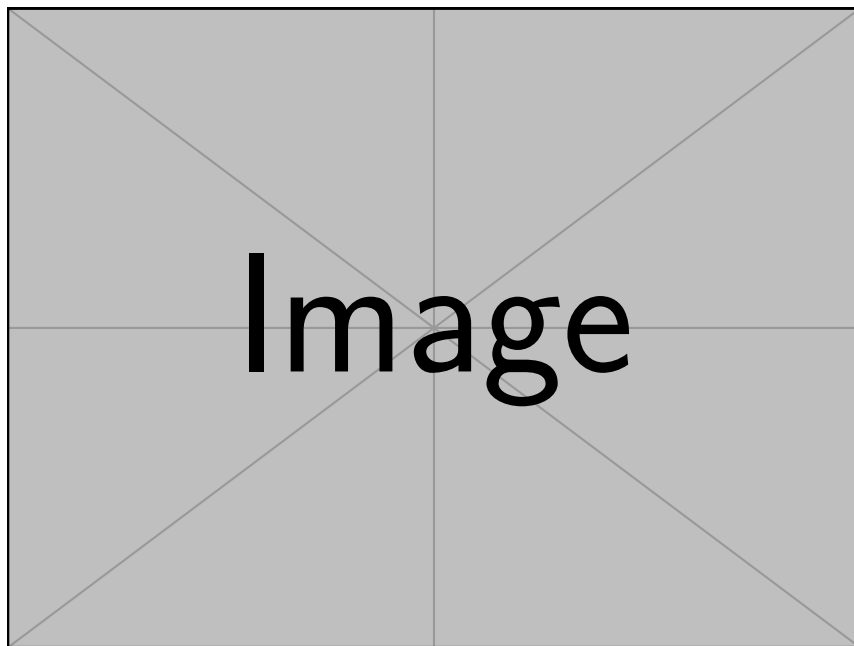


图 A.1 附录中的图片

### B 另一个附录