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

图目录

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。

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。

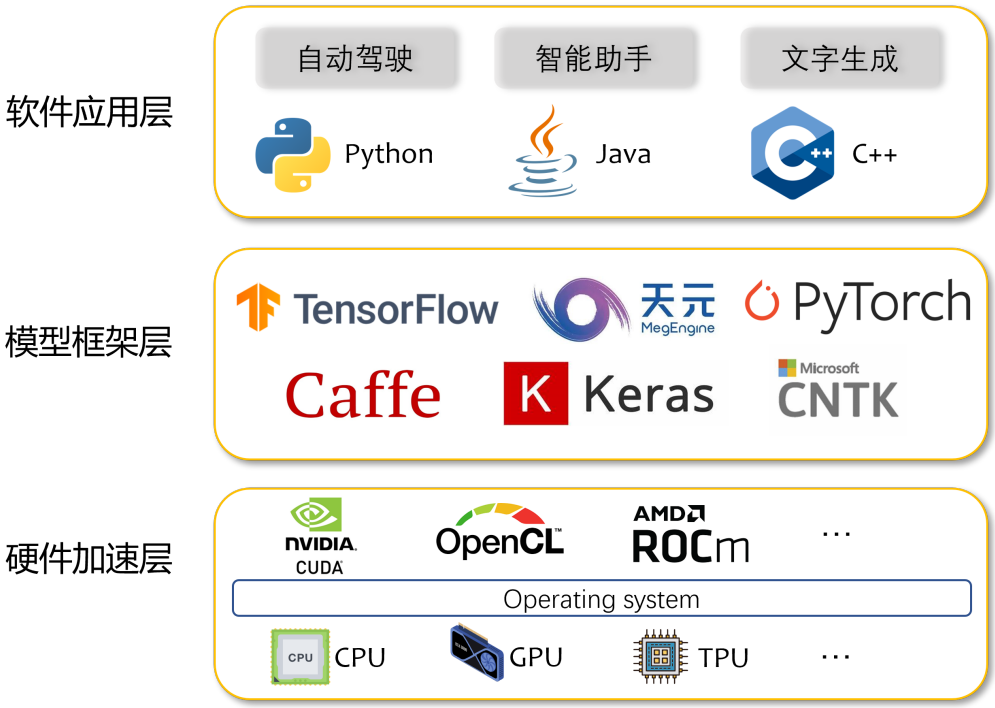

图 1.1 AI 系统架构图

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[11]。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[12]。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[13]。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[14]。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[15]。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[16]。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[17]。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[18]。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[19]。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[20]。

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 AI 系统的模型框架层，研究者逐渐意识到预训练模型的本身及其其所依赖的运行框架和算子在 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，该平台上累计发布的开源模型数量已超过 200 万个[21]。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，开源模型库正逐步成为攻击者投放恶意载荷的新型渠道。JFrog 于 2024 年 2 月发布的分析报告指出，其在 Hugging Face 平台上发现了超过 100 个恶意模型，涉及 TensorFlow、PyTorch 等多个主流深度学习框架。这些模型在加载或推理阶段可触发反向 shell、任意文件读写、启动特定程序以及代码执行等恶意行为[22]。相较于传统的软件包投毒攻击，模型与框架层面的攻击更贴近模型的实际执行路径，能够自然嵌入正常的模型加载与推理流程中，因而通常具备更强的隐蔽性和更高的潜在危害性。

围绕模型框架层的安全风险，现有研究已从多个角度展开系统性探索，相关工作大体可归纳为恶意模型行为分析、模型安全检测机制以及模型框架层漏洞挖掘等方向。从攻击目标与实现方式的角度看，模型层面的恶意逻辑注入主要可以划分为两类。

第一类是传统机器学习语境下的恶意模型，其核心目标在于操纵模型的预测或决策结果，而非直接执行系统级恶意行为。例如，攻击者可通过精心设计的训练过程，使智能驾驶模型在特定条件下将红灯错误识别为绿灯，从而间接诱发交通事故。这类攻击主要关注模型推理行为本身的安全性，对系统执行环境的影响通常是间接的。代表性研究包括后门攻击，即在训练阶段向模型中植入隐蔽触发器，使模型在正常输入下表现正常，而在触发条件出现时输出攻击者预期结果[23-25]；以及对抗样本攻击，通过对输入样本施加微小扰动诱导模型产生错误分类[26-28]。近年来，随着大参数模型高效微调技术的发展，研究者进一步发现，可利用 LoRA 等轻量化微调机制在不显著影响模型整体性能的前提下植入恶意触发逻辑，从而实现更加隐蔽的攻击[29-30]。

第二类则是将 AI 模型本身作为恶意逻辑载体的攻击方式。在这一语境下，模型不再仅用于产生错误预测结果，而是被直接用于承载、隐藏并触发恶意软件或恶意代码，从而对运行模型的系统环境造成实质性威胁。现有研究表明，此类攻击主要通过以下三种方式实现。其一，攻击者将恶意软件或恶意逻辑嵌入模型的二进制参数或特定层次结构中，并在模型运行阶段对恶意载荷进行重组与触发。Hua 等人提出的 Malmodel 技术[31]，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[32]，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[33]、Evilmodel 2.0[34]以及 StegoNet[35]，则采用最低有效位 (Least Significant Bit，LSB) 隐写术将恶意软件隐藏于模型权重中。其二，攻击者将恶意逻辑直接嵌入模型的 lambda 层中。这类攻击主要适用于支持 lambda 层的模型框架（如 TensorFlow），通过在模型执行过程中触发任意代码执行实现攻击。然而，该方式通常较易被检测，因为仅需检查模型中是否存在 lambda 层并分析其逻辑即可识别异常行为[36-37]。其三，也是目前最为普遍的一类方式，是利用 pickle 等不安全的模型序列化格式，将恶意逻辑嵌入模型文件中，并在模型反序列化过程中触发代码执行[38-40]。针对这一威胁，工业界已提出多种检测与分析工具。例如，Picklescan 可对 pickle 格式的模型文件进行反序列化分析，从而识别潜在的恶意函数调用[41]；Fickling 提供了对 Python pickle 对象的反编译、静态分析和字节码重写能力，

既可用于检测嵌入 PyTorch 模型的恶意行为，也可被用于构造攻击载荷[42]；Picklescan 同样支持对基于 pickle 的恶意 PyTorch 模型进行检测[43]。目前，业界较为先进的模型检测工具包括 Protect AI 公司推出的 ModelScan，该工具能够识别包括基于 pickle 的恶意模型和 TensorFlow lambda 层攻击在内的多种模型级恶意行为[44]。

综上所述，现有研究已从多个角度揭示了模型框架层在 AI 系统供应链中面临的安全风险，充分地证明了模型本身可以被用作攻击载体。然而，这些工作大多将风险归因于恶意模型本身或不安全的序列化机制，从而将模型框架层的安全边界界定在模型层面，这是不完备的，事实上模型框架层自身和为模型框架提供的算子层面的攻击仍未被充分研究。

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 等专用计算设备以满足大规模并行计算与高吞吐推理需求，其底层由 GPU 硬件、设备驱动、运行时库以及 CUDA、OpenCL 等编程框架共同构成。这一层通常被视为“可信的计算基础”，但随着 GPU 架构与软件栈复杂度的持续提升，相关组件逐渐暴露出新的安全风险，使其在 AI 供应链中的角色从单纯的计算加速单元，演变为潜在的攻击入口。

已有研究主要从 GPU 架构与系统安全的角度，对底层机制进行了系统性分析。在内存隔离与地址随机化方面，ASLR 作为现代系统的重要防护机制，已在 CPU 平台上被广泛研究。相关工作系统分析了 Linux、Windows 及 macOS 等平台上的 ASLR 实现缺陷，揭示了其随机化熵不足、旁路泄漏及错误配置等问题，并提出了多种绕过手段。然而，相较于成熟的软件系统，GPU 侧的地址空间布局与随机化机制长期缺乏公开文档，相关研究极为有限，使得 GPU 在 AI 系统中的实际隔离能力仍不清晰。

围绕 GPU 的安全性，现有研究进一步从侧信道攻击与内存漏洞两个方向展开。在侧信道方面，研究者通过对 GPU 缓存层次结构、Tensor Core 以及互连机制的逆向分析，揭示了跨进程甚至跨租户的信息泄漏风险；在内存安全方面，多项工作表明 GPU 内核中同样存在栈溢出、越界访问及未初始化内存使用等问题，可被用于数据泄漏或控制流劫持。这些研究表明，GPU 并非一个天然隔离、可信的执行环境，其底层缺陷可能被攻击者利用，从而影响上层 AI 应用的安全性与可靠性。

从供应链视角来看，上述研究虽主要聚焦于单一系统或攻击技术本身，但其揭示的 GPU 架构缺陷与实现漏洞，实际上构成了 AI 系统硬件加速层的重要供应链风险。一旦底层 GPU 运行时或驱动存在可被稳定利用的安全弱点，上层依赖该硬件执行的模型推

理、训练任务乃至云端 AI 服务，均可能受到连锁影响。尤其是在模型即服务（Model-as-a-Service）与多租户 GPU 共享场景下，硬件层面的安全问题可能通过框架与运行时逐级放大，成为 AI 供应链中难以察觉却影响深远的薄弱环节。

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/OL]//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/OL]. 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/OL]//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. Austin, USA: 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/.
[41] Python Software Foundation. pickletools —Tools for pickle developers[EB]. 2023.
[42] Of BITS T. Fickling @ DEFCON AI Village 2021[EB]. 2021.
[43] Mmaitre314. Python Pickle Malware Scanner[EB]. 2024.
[44] Protectai. ModelScan: Protection against Model Serialization Attacks[EB]. GitHub. 2024.

附录

A 一个附录

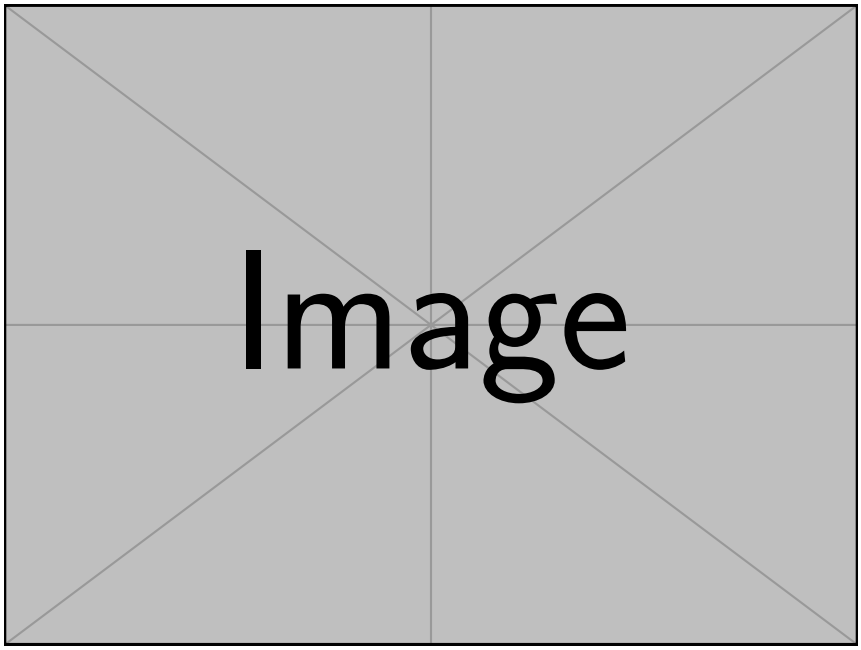

图 A.1 附录中的图片

B 另一个附录