**项目名称： Node.js开源软件仓库安全漏洞与演化趋势分析**

**组长姓名： 赵靖瞳**

**小组成员： 刘浩刚 林子强 王天宇 马千一**

### 一、项目简介

#### 1.1 项目简述

随着开源技术的普及与应用，Node.js作为最受欢迎的JavaScript运行时之一，已广泛应用于各种开发场景。然而，随着技术的不断发展，Node.js的开源生态也面临着诸多安全问题。每年，新的漏洞不断涌现，影响着成千上万的开发者和企业。为了理解这些漏洞的演化趋势，我们开展了“Node.js开源软件仓库安全漏洞与演化趋势分析”项目，旨在通过分析CVE（公共漏洞披露数据库）中的漏洞数据，探讨Node.js的安全漏洞如何随着时间的推移发生变化，及其类型和严重性的分布特征。

本项目首先通过爬虫技术从CVE数据库中收集了2011年至2025年间的Node.js漏洞记录，涵盖了从CVE-2011到CVE-2025之间的多个漏洞数据。数据收集后，进行了漏洞分类，并结合pandas和matplotlib等工具对这些数据进行了详细的统计和可视化。我们不仅分析了漏洞数量的变化趋势，还针对漏洞的类型进行了分类，并对漏洞的未来趋势进行了预测。

此外，本项目还通过线性回归模型对未来几年的漏洞数量进行了预测，以此为技术社区和企业提供可能的安全趋势参考。最终，项目通过一系列图表和可视化展示了漏洞演化的历史数据和预测结果，揭示了Node.js开源生态中的潜在安全风险。

#### 1.2 项目可交付品清单

1. ipynb文件：使用jupyter notebook编写的分析代码，包括数据爬取以及可视化、
2. Csv文件：爬取下来的漏洞数据
3. word文件：攥写的项目报告书

### 二、需求规格说明

项目需求：

本项目旨在分析Node.js开源软件仓库中的安全漏洞及其演化趋势，主要通过收集和分析CVE数据库中的Node.js漏洞数据，探讨漏洞发现的频率变化、漏洞类型的演化趋势以及未来潜在的漏洞走向。具体需求包括：

（1）数据收集：通过爬取CVE数据库，收集Node.js相关的安全漏洞数据。爬取的内容包括每个漏洞的CVE ID、漏洞描述和类型，以及漏洞发生的年份。

（2）数据清洗与处理：对收集到的数据进行清洗，包括对漏洞发布年份进行提取、缺失值处理以及漏洞类型的分类。数据清洗后的数据将用于后续的分析和可视化。

（3）漏洞类型分析：对不同类型的漏洞进行分类，并分析各个漏洞类型在不同年份的分布情况。漏洞类型的分类可以包括：权限提升（Privilege Escalation）、远程代码执行（RCE）、SQL注入（SQL Injection）、跨站脚本攻击（XSS）等。

（4）时间序列分析：对漏洞的发现频率、类型的变化进行时间序列分析，帮助理解漏洞的演化趋势。

（5）趋势预测：基于历史漏洞数据，使用回归分析方法预测未来几年可能出现的漏洞数量和类型。这将为开发者、运维人员和安全专家提供有价值的参考。

（6）可视化展示：使用Matplotlib、Pandas等工具生成可视化图表，帮助用户清晰理解漏洞的变化趋势，包括漏洞数量随时间的变化、不同类型漏洞的分布情况、以及未来漏洞的预测结果。

预期读者：

（1）Node.js开发者和运维人员：本项目提供的漏洞分析与预测将帮助开发者和运维人员了解Node.js开源软件中的安全问题，及时发现潜在的安全隐患。

（2）安全专家与研究人员：安全专家和研究人员可以基于项目的分析结果，进一步研究Node.js的安全性演化，为技术社区提供安全改进的方向。

（3）技术决策者与管理者：对于技术决策者来说，了解漏洞的演化趋势可以帮助制定合理的安全策略，并采取必要的防护措施，减少系统受到攻击的风险。

（4）学术界与教育领域：本项目的分析方法和结果对学术研究和教育课程有一定的借鉴意义，可以作为数据分析与安全研究的案例。

研究方法：

（1）数据收集：通过编写Python爬虫，利用requests和BeautifulSoup库，自动化从CVE网站爬取Node.js相关的安全漏洞数据。数据包括CVE ID、漏洞描述、漏洞类型以及漏洞发布时间。

（2）数据清洗与分类：使用正则表达式提取漏洞名称中的年份，将每个漏洞的年份提取到新的“Year”列中，处理缺失数据并进行填充。漏洞类型使用基于描述的关键字分类方法进行分类，确保每个漏洞都能准确归类。

（3）数据分析：利用Pandas进行数据分析，首先进行各类统计分析（如每年漏洞数量、漏洞类型的分布），然后使用回归分析对未来漏洞趋势进行预测。

（4）时间序列分析与趋势预测：利用线性回归模型，分析历史数据中的漏洞数量随时间变化的趋势，并基于该趋势预测未来几年的漏洞数量。

（5）数据可视化：使用Matplotlib对清洗后的数据进行可视化，展示漏洞数量变化、漏洞类型的分布情况以及预测的趋势线，帮助用户更加直观地理解数据。

（6）结果呈现：通过图表展示每年发布的漏洞数量、不同类型漏洞的数量，以及未来漏洞的预测数量。同时，利用回归模型的结果，展示未来几年漏洞趋势的变化，帮助相关人员作出及时的安全决策。

### 五、项目总结

**5.1 Node.js评估**

综合来看，Node.js 作为一种高效的开发平台，其在 I/O 密集型应用和实时应用领域表现出色，但它在安全性方面依然面临不少挑战。随着技术的不断发展和社区的积极参与，Node.js 的安全性也在逐步改进。根据研究表明，虽然 Node.js 在持续修复已知漏洞，但随着使用范围的扩大，新漏洞不断出现，漏洞管理和安全性将是未来发展的重点。

为了更好地应对这些挑战，开发者不仅需要跟进 Node.js 和其依赖模块的最新安全更新，还需要在开发初期就构建起有效的安全框架，防止潜在的安全问题影响应用的稳定性和安全性。

**5.2 总结与展望**

本项目深入分析了 Node.js 的开源软件仓库中的安全漏洞以及其演化趋势。通过从 CVE 网站爬取并分类 Node.js 相关漏洞数据，结合数据分析和可视化方法，揭示了 Node.js 在过去几年中的漏洞情况及其潜在的安全隐患。此项目不仅揭示了 Node.js 的安全问题，也为未来的安全修复和漏洞预防提供了重要参考。

通过线性回归模型，预测了未来几年 Node.js 漏洞的趋势。根据分析结果，Node.js 的漏洞数量将在未来几年内持续增长，尤其是在 2026 至 2030 年之间，预计漏洞数量会大幅增加。虽然这种增长趋势难以避免，但通过及时更新和修复漏洞，社区可以减缓这一趋势的加剧。

然而，在本项目的过程中，也有一些不足之处：

（1）漏洞数据来源单一：虽然 CVE 数据库为我们提供了一个很好的漏洞数据源，但由于其更新滞后以及数据的局限性，未来的漏洞预判可能会存在误差。如果能整合更多来自不同平台和安全机构的数据，将使预测结果更加准确。

（2）漏斗预测模型的简化：虽然使用了线性回归模型进行漏洞预测，但这一方法相对简单，未来可以尝试引入更复杂的机器学习算法，如时间序列分析模型，以提高预测的准确性和可靠性。