Beijing Forest Studio

北京理工大学信息系统及安全对抗实验中心

源代码漏洞分类

硕士研究生 谢宁

2023年11月26日

问题回溯

- 总结反思
  - 控制演讲的语速，特别是讲述重点时需要放缓
  - 明确哪些概念需要重点解释，且要基于基础概念解释
  - 声音要有起伏，注意强调重点
  - 附录格式使用不当

- 相关内容
  - 2023.05.14 孔令迪《源代码漏洞检测》
  - 2023.03.26 谢宁《软件漏洞检测及其严重性评估》
  - 2022.10.30 孔令迪《函数级漏洞检测》

内容提要

- 预期收获
- 题目内涵解析
- 研究背景与意义
- 研究历史与现状
- 知识基础
- 算法原理
  - VulExplainer
- 特点总结与工作展望
- 参考文献

背景简介

- 预期收获
  - 掌握源代码漏洞分类基本概念
  - 了解基于CWE类型的漏洞数据划分方法
  - 了解基于CodeBERT模型的源代码漏洞分类方法

内涵解析与研究目标 源代码漏洞分类

- 源代码漏洞
  - 含漏洞源代码：含有漏洞的一个函数或一整个文件
  - 含漏洞代码提交：被提交到Github或其它网站中包含漏洞的代码提交
  - 含漏洞补丁：对漏洞部分进行修补的代码提交或源代码
- 分类对象
  - CWE类型：由MITRE公司维护，并提供描述漏洞的通用语言和分类法
  - 商业类型：从商业需求或者软件需求出发设定的漏洞类型
  - 自定义类型：根据研究需要或者从作者观点出发设定的类型
- 研究目标
  - 面向漏洞分析领域的漏洞评估
  - 研究CWE类型漏洞分类、源代码中的漏洞特征提取等问题

研究背景

- 研究背景
  - 漏洞分类技术广泛应用于漏洞严重性评估、漏洞检测和漏洞修复等领域
  - 漏洞分类方法对专家知识的依赖性高，分类结果生成周期长
  - 漏洞检测中无法提供更加详细的漏洞类型信息
    - CWE-ID解释检测到的漏洞，使安全分析人员不能完全理解检测到的漏洞

CVE-2022-45380 Detail

Description

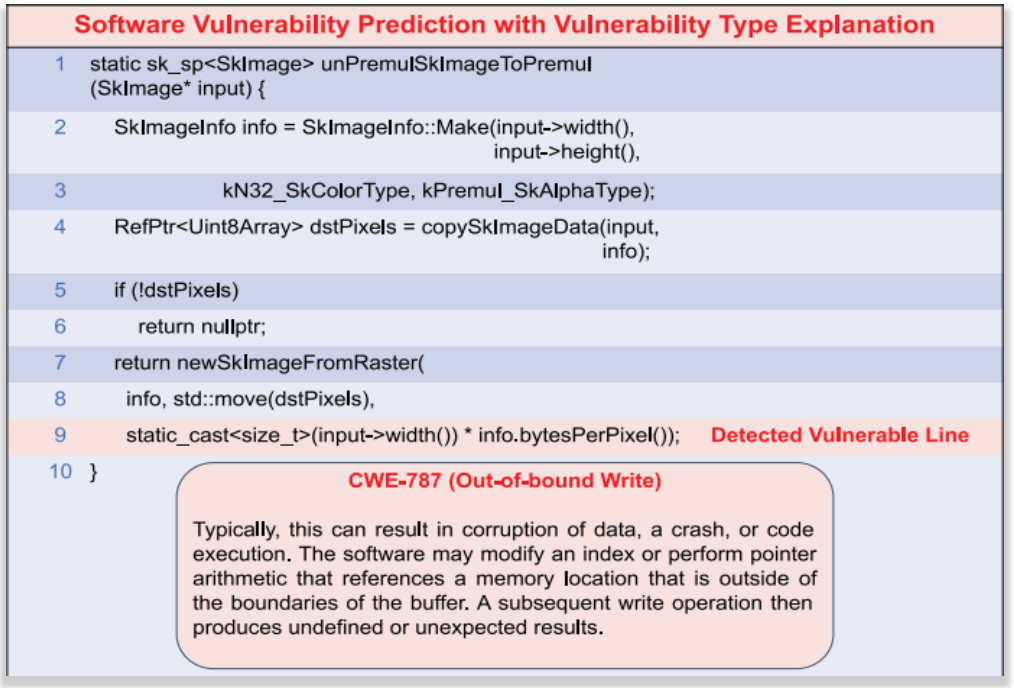
Jenkins JUnit Plugin 1159.v0b_396e1e07dd and earlier converts HTTP(S) URLs in test report output to clickable links in an unsafe manner, resulting in a stored cross-site scripting (XSS) vulnerability exploitable by attackers with Item/Configure permission.

Weakness Enumeration

CWE-ID	CWE Name	Source
CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	NIST

研究意义

- 研究意义
  - 指导漏洞修复：帮助分析检测到的补丁类型，确定补丁的必要性
  - 简化漏洞补救过程：根据CWE类型采取临时缓解措施，简化评估过程
  - 提高NVD提供的元数据质量：帮助补充NVD中百分之三十的低分类精度数据中的内容
  - 减少人力消耗：省去理解和分析漏洞的耗时，避免损害早期补救的价值，减少手工分析带来的延迟，使用自动化的方法将检测到的漏洞分为细粒度的漏洞类型



漏洞分类研究刻不容缓！

研究历史

2006 Li等人分析了bug的特征，并通过文本分类和信息检索技术对bug进行了分类，初步实现了对bug类型进行简单分类

2016 Ni等人提出了一种使用朴素贝叶斯分类器以CVE条目漏洞类型的分类方法，但他们只处理了数据集中最常见的10个CWE-ID

2017 Davari等人提出了一个漏洞自动分类框架，该框架构建f1得分最高的机器学习分类器来标记未知漏洞，并对来自Firefox项目的漏洞数据集进行了测试

2018 Yang等人采用模糊熵进行漏洞特征选择，特征权重由TF-IDF确定，支持向量机（SVM）参数采用改进粒子群算法作为优化算法，只有8个漏洞类别

2020 Chen等人提出了一种基于TF-IGM和特征选择的漏洞严重程度分类框架，特征选择在不同数据集上提高分类性能的有效性很不稳定

2021 Das等人提出了一种新的基于变换的学习框架（V2W-BERT），通过结合自然语言处理、链接预测和迁移学习的思想来获得高度精确的CWEs映射

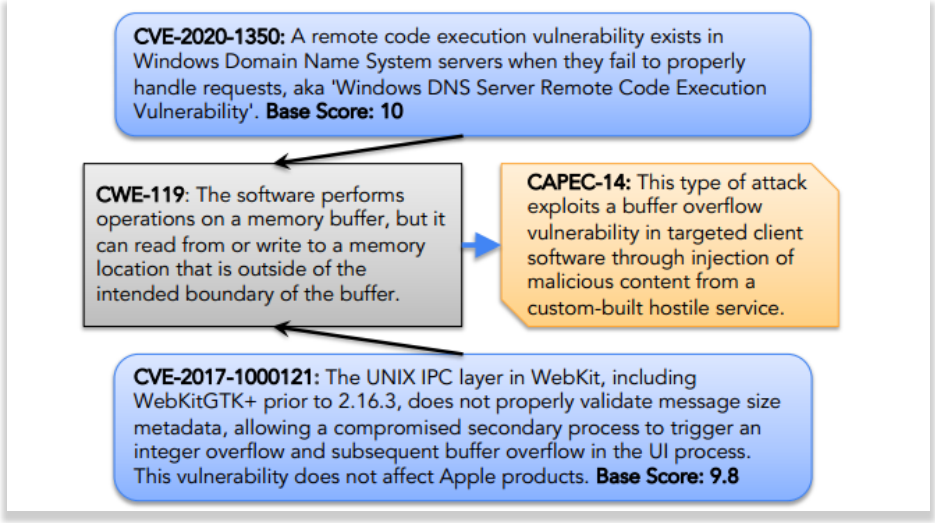
2021 Wang等人基于纯自注意力机制的transformer编解码器，实现了漏洞类型分类。分类方法优于所有常见的文本分类基线机器学习和深度学习算法在前10位的CWE类别上的分类表现较好

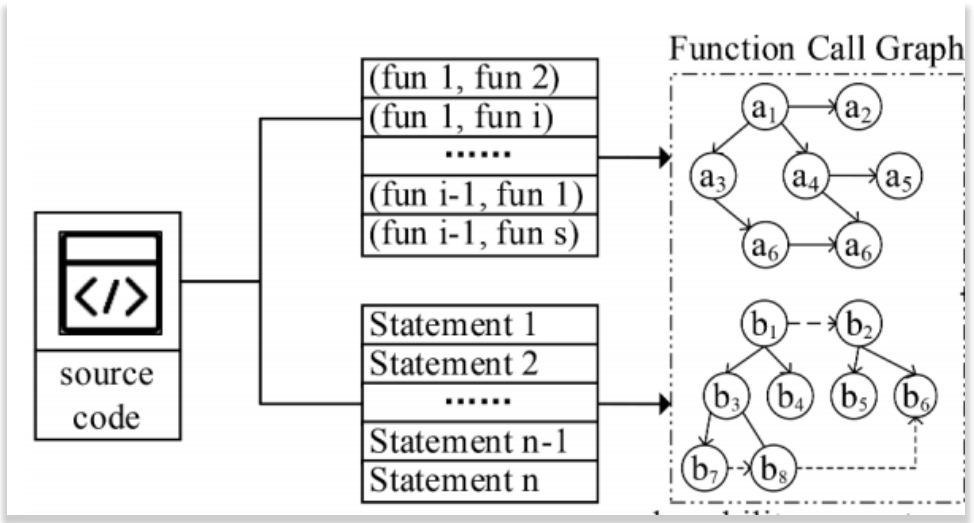
2023 Fu等人提出了一种基于CWE类型和teacher-student学习框架的数据分组方法。将不平衡的标签分布根据CWE类型分层划分为多个子分布，从而准确地预测出所有的CWE-ID

从局部到全面，漏洞分类从仅关注常见类型发展为全类型检测

研究现状

- 漏洞分类方法
  - 流行的漏洞被高度报道，不同软件漏洞数据的分布高度不平衡
- 基于漏洞描述的漏洞分类方法
  - 获取漏洞描述的过程极度依赖于专家知识
- 基于源代码的漏洞分类方法
  - 专注于研究个别漏洞类型的特征（CWE前10），应用局限性较大





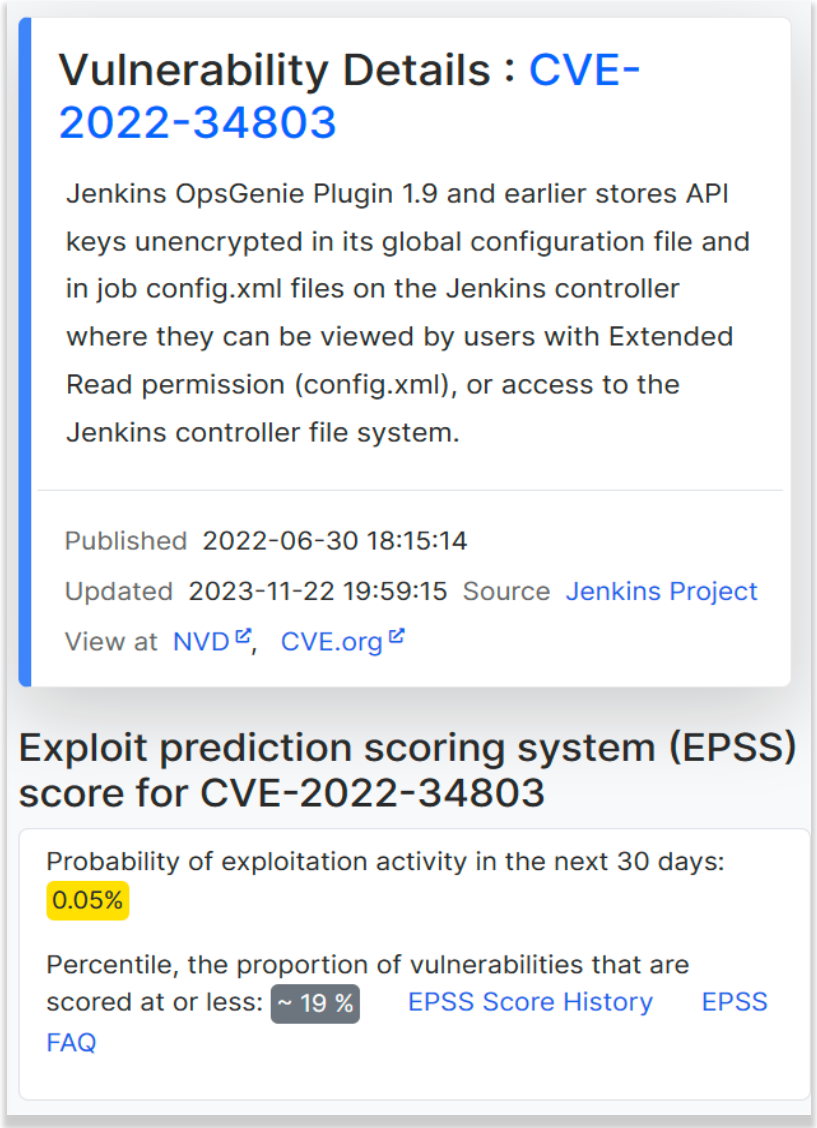
知识基础 CWE类型

- CWE abstract类型
  - Category类型：包含一组共享共同特征的其他条目
    - Category类型不是缺陷，而是帮助用户找到具有所述共同特征的漏洞的结构性项目
  - Class类型：以抽象的方式描述，通常独立于任何特定的语言或技术
    - 比Pillar类型更具体，但比Base类型更抽象
  - Base类型：以抽象方式描述缺陷，但有足够的细节推断检测和预防的具体方法
    - 比Variant类型更抽象，但比Class类型更具体
  - Variant类型：与某种产品相关的缺陷，通常涉及特定的语言或技术
    - 比Base类型更具体
  - Deprecated类型：表示某个特定的漏洞类型或模型已经过时，不被认为是主要威胁

CWE abstract类型是对CWE-ID进行分类的重要依据

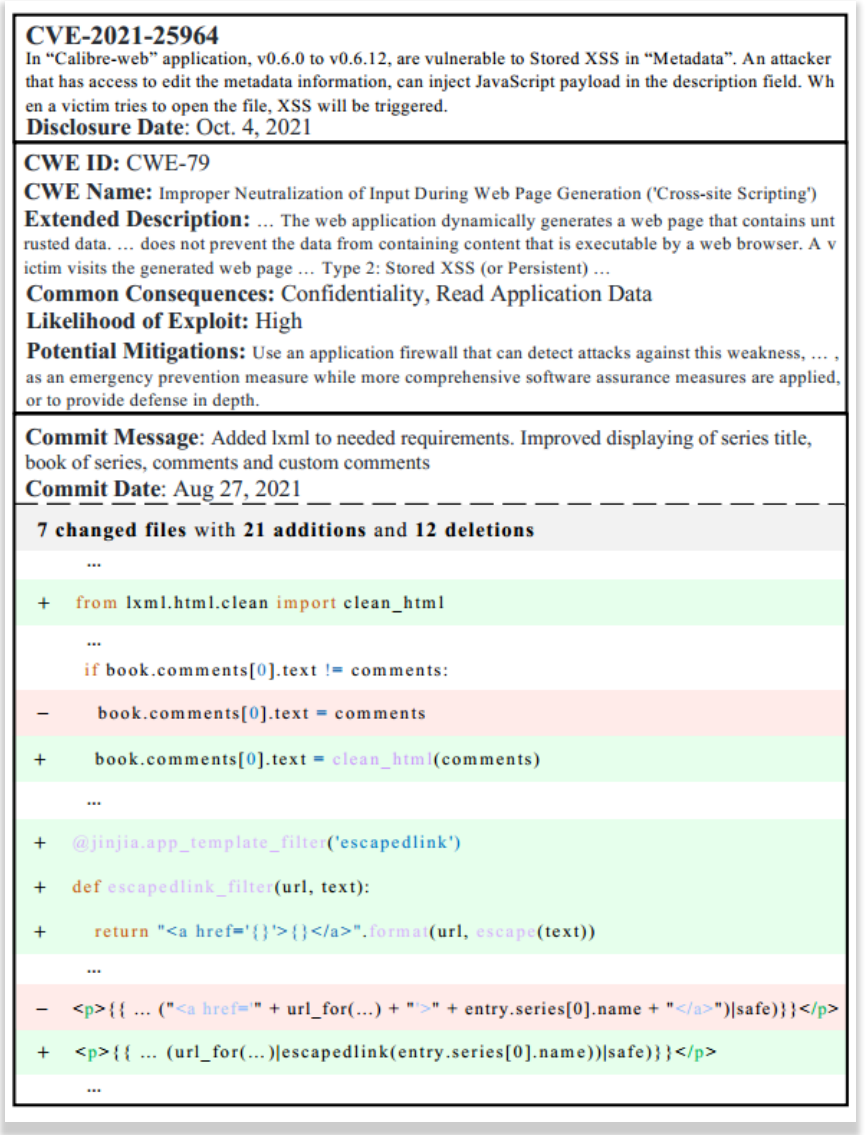
知识基础 漏洞描述的漏洞分类

- 漏洞描述的漏洞分类
  - 对漏洞描述进行分析，确定漏洞类型
    - 以文字描述为基础
  - 特点：适应性和灵活性强
    - 适用于各种不同类型的漏洞，依赖于文字描述而非具体的技术实现
    - 能够适应新漏洞类型的出现，因为只需更新自然语言的算法
  - 局限性
    - 受到作者主观表达的影响
    - 容易随着时间发生概念漂移
    - 分类周期过长



知识基础 漏洞检测的漏洞分类

- 漏洞检测的漏洞分类
  - 结合多种漏洞特征和检测方法分析
    - 不仅能够发现漏洞的存在，还能简单地检测出漏洞的类型
  - 特点：多维度和技术综合
    - 综合考虑漏洞的不同特征
      - 包括但不限于代码审查等
    - 在结合静态分析和动态分析等多种技术手段，使得漏洞检测更全面
  - 局限性
    - 精确度不足，无法覆盖所有漏洞类型
    - 针对性强，需要不断更新和维护



CVE-2021-25964

In "Calibre-web" application, v0.6.0 to v0.6.12, are vulnerable to Stored XSS in "Metadata". An attacker that has access to edit the metadata information, can inject JavaScript payload in the description field. Wh en a victim tries to open the file, XSS will be triggered.

Disclosure Date: Oct. 4, 2021

CWE ID: CWE-79

CWE Name: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

Extended Description: ... The web application dynamically generates a web page that contains unt rusted data. ... does not prevent the data from containing content that is executable by a web browser. A v ictim visits the generated web page ... Type 2: Stored XSS (or Persistent) ...

Common Consequences: Confidentiality, Read Application Data

Likelihood of Exploit: High

Potential Mitigations: Use an application firewall that can detect attacks against this weakness, ... , as an emergency prevention measure while more comprehensive software assurance measures are applied, or to provide defense in depth.

Commit Message: Added lxml to needed requirements. Improved displaying of series title, book of series, comments and custom comments

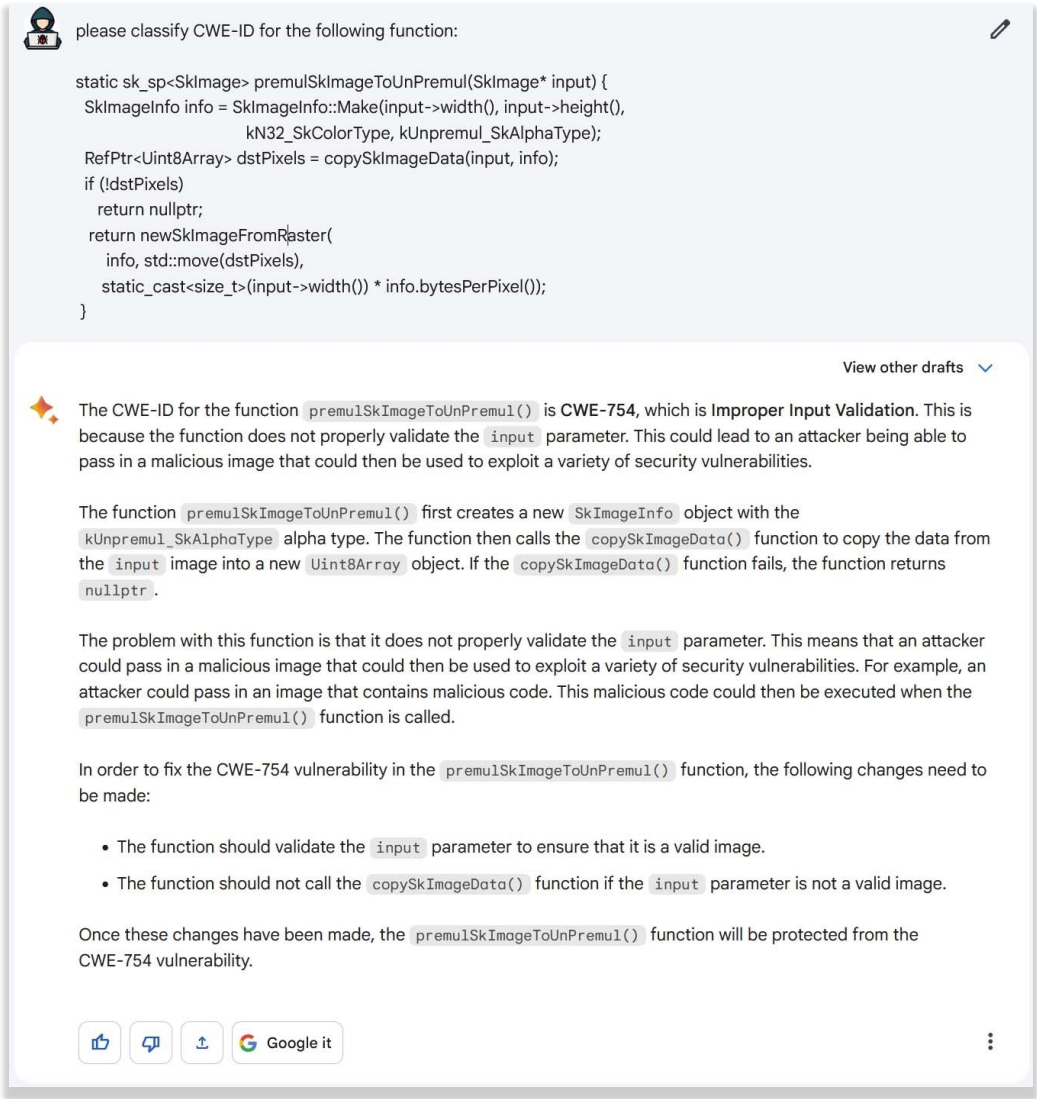
Commit Date: Aug 27, 2021

7 changed files with **21 additions** and **12 deletions**

```
      ...
+   from lxml.html.clean import clean_html
      ...
      if book.comments[0].text != comments:
-       book.comments[0].text = comments
+       book.comments[0].text = clean_html(comments)
      ...
+   @jinjia.app_template_filter('escapedlink')
+   def escapedlink_filter(url, text):
+       return "<a href='{}'>{}</a>".format(url, escape(text))
      ...
-   <p>{{ ... ("<a href='" + url_for(...) + "'>" + entry.series[0].name + "</a>")|safe)}}</p>
+   <p>{{ ... (url_for(...)|escapedlink(entry.series[0].name))|safe)}}</p>
      ...
```

知识基础 大语言模型的漏洞分类

- 大语言模型的漏洞分类
  - 由大语言模型进行漏洞分析
    - 拥有代码分析和生成样板代码的能力
  - 特点：特征提取过程简单
    - 需要较为完整的指定上下文
    - 无法对CWE精确类型进行分类
  - 代码分析、代码生成
  - 局限性
    - 需要专门准备训练模板
    - 数据集要求的数据量大且需预处理



大语言模型主攻代码分析和代码生成

知识基础 数据源

- 数据源说明
  - 数据源名称：Big-Vul
  - 内容：修复前后的漏洞函数、漏洞类型和评分等

vulnerability_classification	ref_link	commit_id	commit_message	files_changed
DoS Exec Code Overflow	https://github.com/bratsche/pango/commit/4de30e5500eaeb49f4bf0b7a07f718e149a2ed5e	4de30e5500eaeb49f4bf0b7a07f718e149a2ed5e	[glyphstring] Handle overflow with very long glyphstrings	{"sha": "8fb70313eb8835dcce812a86209e2a7d88457795", "filename": "pango/glyphstring.c", "status": "modified", "additions": 20, "deletions": 6, "changes": 26, "blob_url": "https://github.com/bratsche/pango/blob/4de30e5500eaeb49f4bf0b7a07f718e149a2ed5e/pango/glyphstring.c", "raw_url": "https://github.com/bratsche/pango/raw/4de30e5500eaeb49f4bf0b7a07f718e149a2ed5e/pango/glyphstring.c", "contents_url": "https://api.github.com/repos/bratsche/pango/contents/pango/glyphstring.c?ref=4de30e5500eaeb49f4bf0b7a07f718e149a2ed5e", "patch": "@@ -61,14 +61,28 @@ pango_glyph_string_set_size (PangoGlyphString *string, gint new_len)\n while (new_len > string->space)\n {\n if (string->space == 0)\n- string->space = 1;\n+ {\n+ string->space = 4;\n+ }\n else\n- string->space *= 2;\n- if (string->space < 0)\n+ {\n+ string->space = G_MAXINT - 8;\n+ const guint max_space = MIN (G_MAXINT, G_MAXSIZE / MAX (sizeof(PangoGlyphInfo), sizeof(gint)));\n+\n+ guint more_space = (guint)string->space * 2;\n+\n+ if (more_space > max_space)\n+ {\n more_space = max_space;\n+\n+ if ((guint)new_len > max_space)\n+ {\n+ g_error (\"%s: failed to allocate glyph string of length %i\\n\",\n+ G_STRLOC, new_len);\n+ }\n+\n string->space = more_space;\n }\n "}

Features	Column Name in the CSV
Access Complexity	access_complexity
Authentication Required	authentication_required
Availability Impact	availability_impact
Commit ID	commit_id
Commit Message	commit_message
Confidentiality Impact	confidentiality_impact
CWE ID	cwe_id
CVE ID	cve_id
CVE Page	cve_page
CVE Summary	summary
CVSS Score	score
Files Changed	files_changed

VulExplainer: A Transformer-Based Hierarchical Distillation for Explaining Vulnerability Types

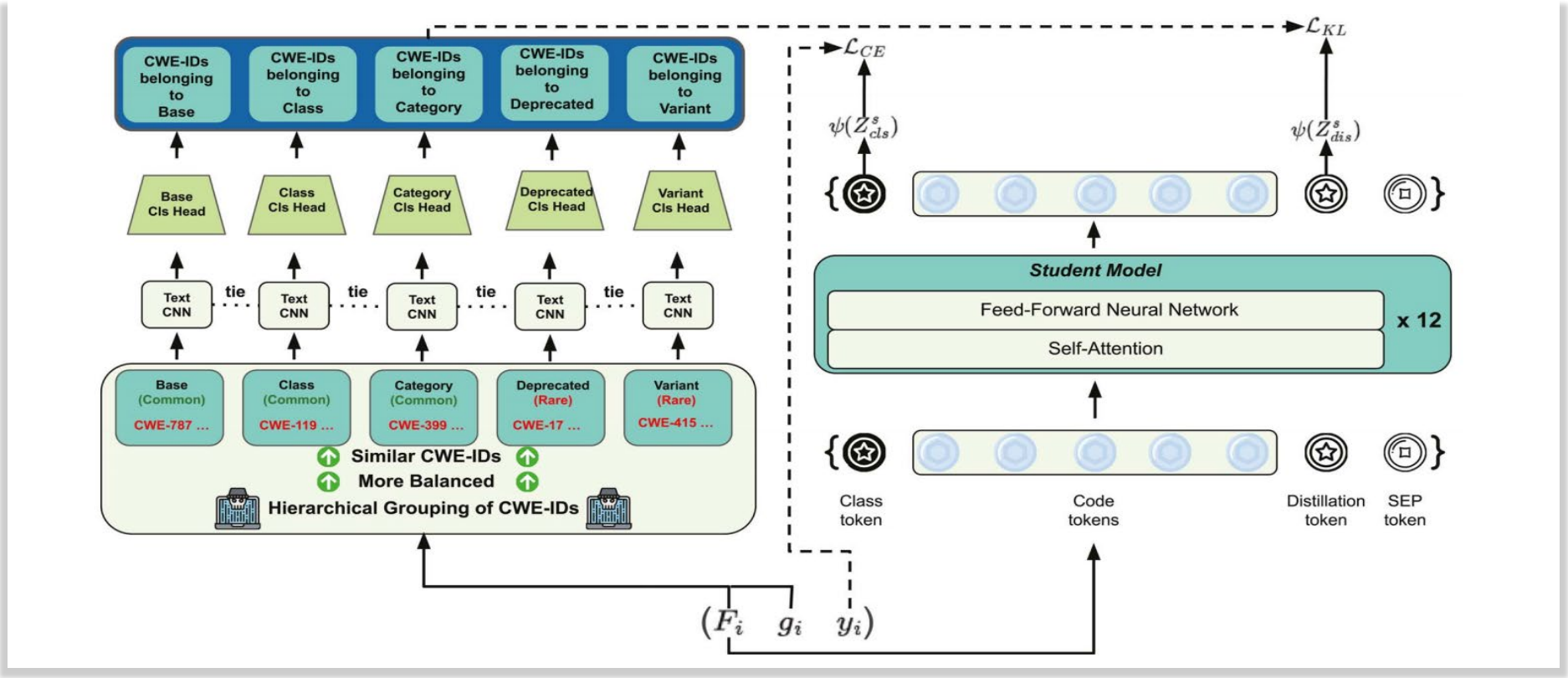
VulExplainer TIPO

T	目标	实现一个漏洞类型解释方法
I	输入	CWE类型的含漏洞源代码 * N
P	处理	1. 将源代码分成具有相同CWE abstract类型的组 2. 对每个CWE abstract类型组训练一个TextCNN-Teacher模型 3. 基于分层Transformer模型进行知识蒸馏
O	输出	漏洞的CWE类型 * 927

P	问题	漏洞分类的精度不足，无法确定给定预测对应于哪种类型的漏洞
C	条件	代码类型为C++
D	难点	降低由于CWE-ID中标记的样本数量高度不平衡导致的漏洞类型预测精度产生的损失
L	水平	IEEE Transactions on Software Engineering 2023（SCI 一区）

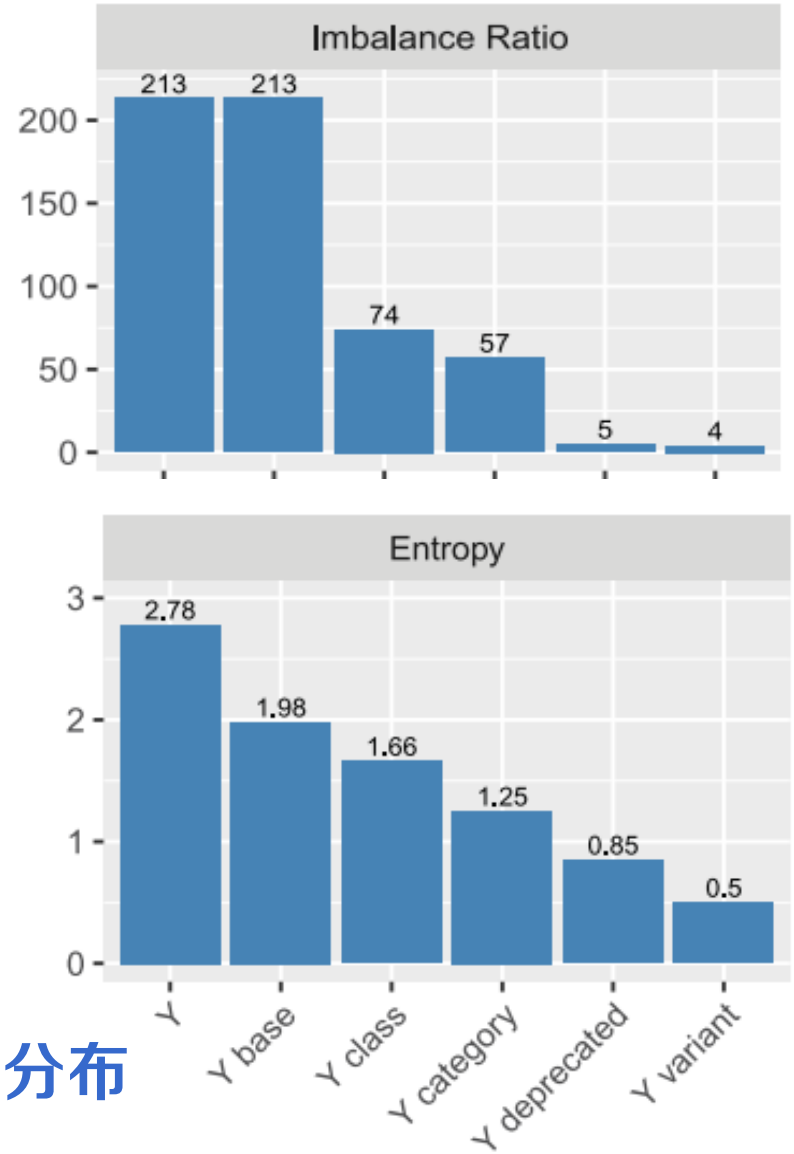
VulExplainer 算法原理图

- 算法原理图
  - 漏洞分组：将源代码分组为具有相同CWE abstract类型的组
  - Teacher模型训练：训练多个TextCNN-Teacher，预测特定类型下的CWE-ID
  - 知识蒸馏：分层提取从训练的多个不同Teacher模型中获得的知识



创新分析 漏洞分组

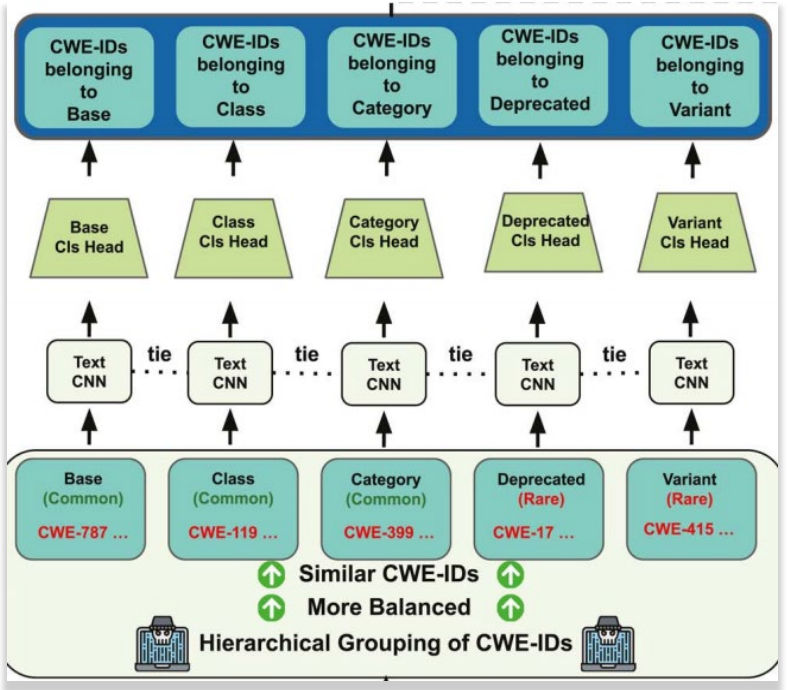
- 分组依据
  - 目标：对抗不平衡的标签分布，将一个复杂/不平衡的数据分布简化为多个简单/平衡的数据分布
  - 类别
    - 五种常见的CWE abstract类型，Class、Base、Category、Variant和Deprecated
  - 内涵：每种类型都提供对公共弱点枚举（CWE）系统中漏洞的不同方面的有价值的见解



依据CWE abstract类型，划分出多个更平衡的子分布

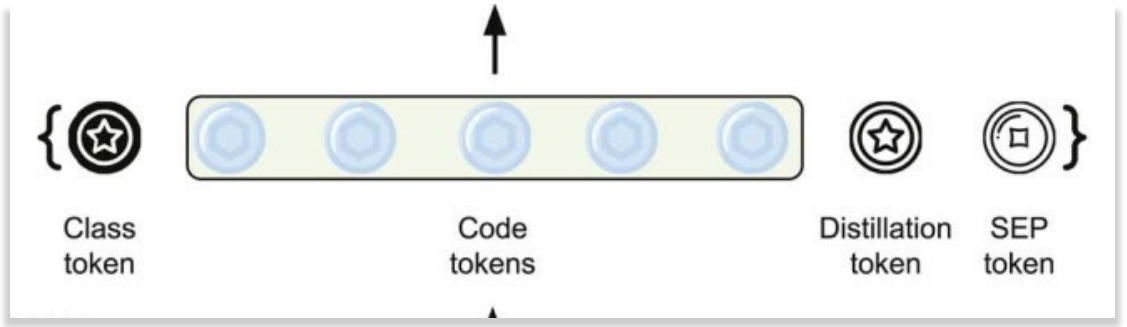
创新分析 Teacher模型训练

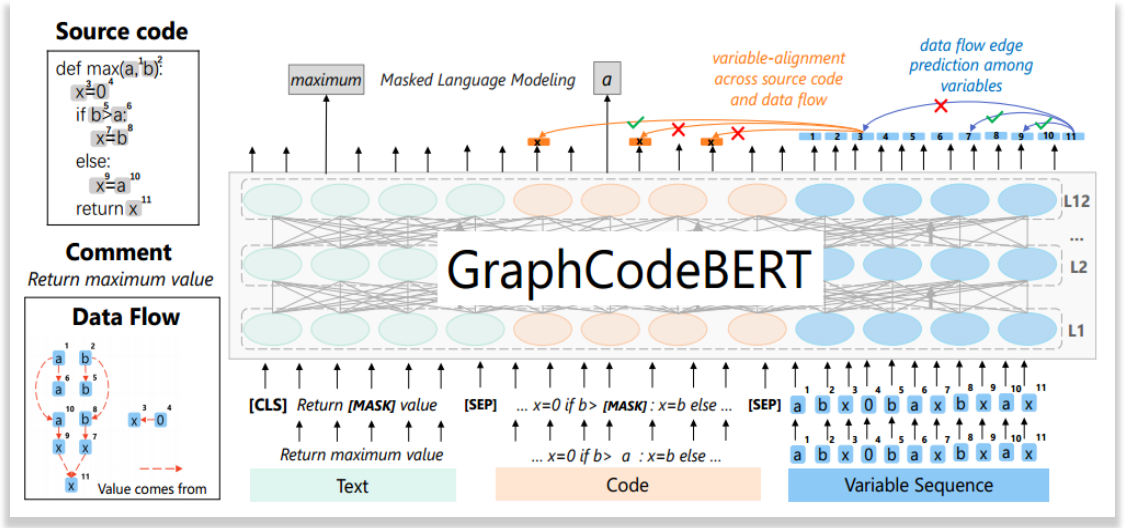
- 数据集准备
  - 对CWE abstract类型进行分组，得到相似且更平衡的CWE-ID组成的组
    - 每组中由相似特征的CWE-ID组成，因此DL模型更容易从中学习
    - 相似类型的CWE-ID构成的数据组，数据分布更加平衡
- TextCNN-Teacher模型构建
  - 对于每一个CWE abstract类型对应的组，训练一个TextCNN-Teacher模型
  - 共享TextCNN-Teacher模型的骨干
  - 对于数据集，构建5个分类头，分别对应于YBase、YCategory、YClass、YVariant和YDeprecated



创新分析 知识蒸馏

- 嵌入Teacher知识
  - 构建漏洞函数的DFG（数据流图）利用python中的tree-sitter包
  - 生成token序列
    - 表示为$F_i = [t_1, \ldots, t_n]$
    - 由n个通过BPE算法分割的token组成
  - 生成标记序列
    - 截断并填充，以使$n = 512$个token
    - [cls]标记：用于学习输入函数的表示，由分类头用于分类 CWE-ID
    - [dis] 标记：用于从Teacher模型中提炼知识（即从TextCNN-Teacher的输出中学习）

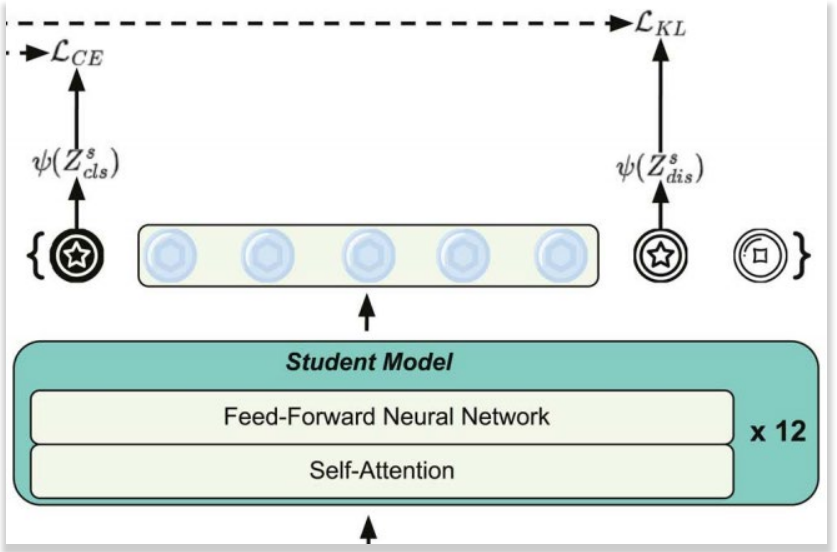




- 分层蒸馏
  - H_0经过12层双向自关注的BERT编码器学习源代码的表示
    - H_0表示为GraphCodeBERT嵌入层的隐藏向量输出
  - E_n将H_{n-1}作为自注意运算的输入，E_n由一个多头自注意操作和2层前馈神经网络组成

$$H_n = E_n(H_{n-1}), n \in \{1, \ldots, 12\}$$

  - 最后一个隐藏层计算出Z^s_{cls}和Z^s_{dis}
    - 其中，H_{12}由H_{cls} 和H_{dis}组合的token产生



- 标签蒸馏
  - $\psi(Z^t)$为教师模型的softmax输出，y_t为真实标签

$$L_{soft} = (1 - \lambda)L_{CE}(\psi(Z^s_{cls}), y) + \lambda L_{KL}(\psi(Z^s_{dis}), \psi(Z^t))$$

$$L_{hard} = (1 - \lambda)L_{CE}(\psi(Z^s_{cls}), y) + \lambda L_{CE}(\psi(Z^s_{dis}), y_t)$$

实验设计 数据资源

- 数据资源
  - 数据集：Big-Vul
- 对比方法
  - CodeBERT，GraphCodeBERT，CodeGPT
  - 数据划分方式：BAGS和LFME
  - 漏洞分类模型
    - Devign（Zhou et al. 2019）
    - ReGVD（Nguyen et al. 2022）
- 实验设置
  - 超参数遵循原作者指定的最佳设置
- 评价指标
  - 准确率（Accuracy，Acc）
  - 总体准确率（Overall Accuracy，Overall Acc）

availability_impact	cve_id	cve_page	cwe_id	access_complexity	confidentiality_impact	integrity_impact
Partial	CVE-2009-1194	https://www.cvedetails.com/cve/CVE-2009-1194/	CWE-189	Medium	Partial	Partial

Models	Optimizer	LR	Grad Clip	Batch	Epoch	λ
Teacher	AdamW	5e-3	1.0	128	50	-
Student	AdamW	2e-5	1.0	16	50	0.7

实验设计 对比实验

- 实验结果
  - 与其他方法相比，VulExplainer方法显著提升ACC，优于所有基线
  - 分层蒸馏法进一步提高BERT的性能
  - 分组策略可以在对相似漏洞类型进行分组时减轻数据的不平衡

Method	Group By CWE Abstract Types					Indicators	
Subsets	Y_{class}	Y_{base}	$Y_{category}$	$Y_{variant}$	$Y_{deprecated}$	Overall Acc	F1
Devign	58.11	44.05	45.10	31.71	38.46	51.16	48.71
ReGVD	60.42	55.95	56.21	36.59	57.69	57.52	56.45
CodeBERT	68.00	58.93	60.78	39.02	57.69	63.19	43.07
CodeGPT	65.26	60.12	64.05	51.22	53.85	63.08	62.30
GraphCodeBERT	63.16	63.19	64.05	41.46	61.54	62.27	62.74
BAGS	63.58	54.17	54.90	51.22	42.31	58.91	57.32
LFME	65.47	58.33	61.44	39.02	50.00	61.57	60.15
GraphCodeBERTSoft–VULEXPLAINER	66.53	64.29	62.75	56.10	57.69	64.58	63.91
CodeBERTSoft–VULEXPLAINER	68.00	64.29	67.97	48.78	61.54	66.09	62.93
CodeGPTSoft–VULEXPLAINER	68.63	62.50	60.78	56.10	57.69	65.05	63.77

实验设计 对比实验

- 实验结果
  - 与其他方法相比，方法为所有基于Transformer的模型实现最佳性能
  - FL和LA方法都没有进一步提高原始准确率
  - 方法在罕见和常见的CWE-ID上都能取得很好的性能

Method	Group By CWE Abstract Types					Indicators	
Subsets	Y_{class}	Y_{base}	$Y_{category}$	$Y_{variant}$	$Y_{deprecated}$	Overall Acc	F1
GraphCodeBERT	63.16	63.69	64.05	41.46	61.54	62.27	62.75
GraphCodeBERT$_{FL}$	64.21	62.50	58.17	53.66	57.49	62.04	61.61
GraphCodeBERT$_{LA}$	63.58	64.29	60.78	43.90	65.38	62.27	62.74
GraphCodeBERT VULEXPLAINER	66.53	64.29	62.75	56.10	57.69	64.58	63.91
CodeBERT	68.00	58.93	60.78	39.02	57.69	63.19	43.07
CodeBERT$_{FL}$	64.63	58.93	66.67	41.46	57.69	62.62	44.42
CodeBERT$_{LA}$	68.84	66.69	60.13	53.66	65.38	66.09	51.64
CodeBERT – VULEXPLAINER	68.00	64.29	67.97	48.78	61.54	66.09	62.93
CodeGPT	65.26	60.12	64.05	51.22	53.85	63.08	62.30
CodeGPT$_{FL}$	63.16	60.71	64.71	51.22	46.15	61.81	61.06
CodeGPT$_{LA}$	62.32	59.52	59.48	58.54	57.69	61.00	61.47
CodeGPT – VULEXPLAINER	68.63	62.50	60.78	56.10	57.69	65.05	63.77

实验设计 对比实验

- 实验结果
  - 分组方法
    - 基于漏洞类型的分层分组策略对所有使用的模型都达到最佳
  - Teacher模型
    - 对于所有使用的模型，TextCNN-Teacher提取知识的能力优于Transformer的Teacher模型
  - 蒸馏方法
    - 对于所有使用的的模型，软蒸馏（本方法）比硬蒸馏产生更好的结果

本方法的所有模块均有最优效果

Compare Grouping Methods	Models	Accuracy	Improvement
CWE Grouping (ours)	GCB	64.58	+4%
Label Freq Grouping	GCB	62.27	-
CWE Grouping (ours)	CB	66.09	+5%
Label Freq Grouping	CB	62.73	-
CWE Grouping (ours)	GPT	65.05	+4%
Label Freq Grouping	GPT	62.27	-
Compare Teacher Methods	**Models**	**Accuracy**	**Improvement**
TextCNN Teacher (ours)	GCB	64.58	+0.4%
Transformer Teacher	GCB	64.35	-
TextCNN Teacher (ours)	CB	66.09	+6%
Transformer Teacher	CB	62.27	-
TextCNN Teacher (ours)	GPT	65.16	+0.2%
Transformer Teacher	GPT	65.05	-
Compare Distil Methods	**Models**	**Accuracy**	**Improvement**
Soft Distillation (ours)	GCB	64.58	+4%
Hard Distillation	GCB	62.27	-
Soft Distillation (ours)	CB	66.09	+5%
Hard Distillation	CB	62.85	-
Soft Distillation (ours)	GPT	65.05	+0.7%
Hard Distillation	GPT	64.58	-

实验设计 其他实验

- TextCNN-Teacher模型作用验证
  - 在整个测试集上达到77%的最佳综合性能
  - 每个Teacher教师在其指定的CWE abstract类型上表现最佳
- 层次知识提炼方法作用验证
  - CodeBERT模型在整个蒸馏过程中获得准确的知识
  - 在保持初始模型性能的完整性和可靠性方面表现好
  - 在解决原始CodeBERT模型所做的不正确预测方面效果良好

Method	Group By CWE Abstract Types				
Subsets	Y_{class}	Y_{base}	$Y_{category}$	$Y_{variant}$	$Y_{deprecated}$
TextCNN Teacher	72.21	77.38	83.66	95.12	88.46
CodeBERT VULEXPLAINER	68	64.29	67.97	48.75	61.54
CodeBERT w/o pre-training	68	58.93	60.78	39.02	57.69
GraphCodeBERT VULEXPLAINER	60.63	47.02	55.56	34.15	34.62

Testing Subset	Methods	Accuracy
CNNT correctly predicted	CB_{VulExp}(ours)	79.52(528/664)
	CB	71.23(473/664)
CB correctly predicted	CB_{VulExp}(ours)	91.85(462/503)
CB wrongly predicted	CB_{VulExp}(ours)	30.19(109/361)

算法总结 VulExplainer

- 算法流程
  - 根据CWE abstract类型将CWE-ID标签分布Y拆分为多个子分布，对相似的CWE-ID进行分组
  - 构建多个TextCNN-Teacher模型，每个模型都在相同的CWE abstract类型中预测CWE-ID
  - 输入多个TextCNN-Teacher模型的知识到基于Transformer的模型中进行分层蒸馏
- 算法优势
  - 以五种CWE的类型为分类依据，划分多个更平衡的子分布，学习到更准确的模型
  - 利用基于Transformer模型自关注机制的精馏方法，获得更高的分类准确率
- 算法不足
  - VulExplainer方法在其他数据集
  - VulExplainer方法不一定推广到其他Transformer模型

特点总结与未来展望

特点总结与未来展望

- VulExplainer
  - 利用CWE abstract类型作为数据划分依据，将CWE-ID标签分布Y拆分为多个子分布，获得多个CWE-ID相似的分组
  - 使用Student-Teacher架构，分类的精确度高
  - 由于数据集限制，适用范围受限
- 未来发展
  - 扩展到其他与漏洞相关的组件
    - 漏洞严重性评估、漏洞优先级划分
    - 提供一体化的解决方案，在整个漏洞生命周期中自动化细粒度漏洞类型分析
  - 注重对系统和应用程序行为的分析
  - 注重协同和共享信息
    - 建立更强大的漏洞数据库和信息共享平台，迅速地响应新发现的漏洞且更好地共享信息

预期收获 回顾分析

- 预期收获
  - 掌握源代码漏洞分类基本概念
    - 利用源代码的代码特征作为分类依据
    - CWE类型：Class、Base、Category、Variant和Deprecated
    - 源代码漏洞分类对专家知识无要求
  - 了解基于CWE类型的漏洞数据划分方法
    - 目标：划分出多个简单/平衡的数据分布
    - 依据：五种CWE分类方式
    - 内涵：相似的CWE-ID数据量接近且特征相似
  - 了解基于CodeBERT模型的源代码漏洞分类方法
    - 数据划分：获得更加平衡的子分布
    - 知识蒸馏：将多种类型的CWE的知识汇总到一个模型中

参考文献

[1] Fu M, Nguyen V, Tantithamthavorn C K, et al. VulExplainer: A Transformer-based Hierarchical Distillation for Explaining Vulnerability Types[J]. IEEE Transactions on Software Engineering, 2023.

[2] Pan S, Bao L, **a X, et al. Fine-grained Commit-level Vulnerability Type Prediction by CWE Tree Structure[C]. ACM 45th International Conference on Software Engineering (ICSE). IEEE, 2023: 957-969.

[3] Nguyen V A, Nguyen D Q, Nguyen V, et al. ReGVD: Revisiting graph neural networks for vulnerability detection[C]. Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings. 2022: 178-182.

[4] Wang Q, Gao Y, Ren J, et al. An automatic classification algorithm for software vulnerability based on weighted word vector and fusion neural network[J]. Computers & Security, 2023, 126: 103070.

知人者智，自知者明。胜人者有力，自胜者强。知足者富。强行者有志。不失其所者久。死而不亡者，寿。

谢谢！

附录A GraphCodeBERT介绍

- **GraphCodeBERT介绍**
  - 将H_0表示为GraphCodeBERT嵌入层的隐藏向量输出
  - H_0经过12层双向自关注的BERT编码器学习源代码的表示
    - $H_n = E_n(H_{n-1})\ n \in \{1, \ldots, 12\}$
  - 每个编码器E_n由一个多头自注意操作和2层前馈神经网络组成
  - E_n将H_{n-1}作为自注意运算的输入，生成自注意隐藏向量An，其中LN为层归一化，Attn为多头自注意机制
    - $A_n = LN\big(Attn(H_{n-1})\big) + H_{n-1}$
  - A_n经过2层前馈层得到H_n，即E_n生成的最终隐藏向量
    - $H_n = LN(FNN(A_n) + A_n)\ n \in \{1, \ldots, 12\}$