

基于钓鱼工具包指纹识别的钓鱼网站检测方法

孙 奥^{1,2}, 罗 梦^{2†}, 侯碧海³, 杨晓春⁴, 王丽娜¹

1. 空天信息安全与可信计算教育部重点实验室, 武汉大学 国家网络安全学院, 湖北 武汉 430072;

2. 区块链与数据安全全国重点实验室, 浙江大学, 浙江 杭州 310058;

3. 中国联合网络通信有限公司浙江省分公司, 浙江 杭州 310058;

4. 东北大学 软件学院, 辽宁 沈阳 110819

收稿日期: 2025-09-02 † 通信联系人 E-mail: meng.luo@zju.edu.cn

基金项目: 国家重点研发计划(2023YFB3106404); 国家自然科学基金(62302442)

第一作者: 孙 奥, 男, 硕士生, 现从事网络犯罪治理方面的研究。E-mail: sunao828@whu.edu.cn

摘 要: 钓鱼工具包是一种代码资源包, 被广泛用于钓鱼网站的快速、批量部署。通过识别钓鱼网站所用工具包, 可高效检测同一工具包部署的钓鱼网站, 支撑钓鱼网站的精准检测和溯源分析。而现有方法大多依赖人工部署工具包, 且准确率较低。针对上述问题, 提出一种基于钓鱼工具包指纹识别的钓鱼网站检测方法, 该方法无需部署即可提取钓鱼工具包在目录结构、文本和资源上的指纹, 采用不同方法计算指纹相似度。将本文方法与 3 个基线方法进行了对比, 实验结果表明, 在同样不部署工具包的情况下, 本文方法的钓鱼网站检测准确率达 99.5%, 与最佳基线方法相当; 钓鱼工具包识别准确率达 80.5%, 较最佳基线方法提升 26.2 个百分点; 钓鱼工具包家族识别准确率达 85.4%, 较最佳基线方法提升 8.6 个百分点。消融实验表明, 目录结构指纹是核心, 文本与资源指纹能有效互补, 三者融合显著提升工具包及家族识别性能。

关 键 词: 钓鱼网站; 钓鱼工具包; 指纹; 相似度; 树编辑距离; 感知哈希

中图分类号: TP393.08

文献标志码: A

文章编号: 1671-8836(2026)04-0568-11

Phishing Website Detection Method Based on Phishing Toolkit Fingerprinting Identification

SUN Ao^{1,2}, LUO Meng^{2†}, HOU Bihai³, YANG Xiaochun⁴, WANG Lina¹

1. Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, Hubei, China;

2. The State Key Laboratory of Blockchain and Data Security, Zhejiang University, Hangzhou 310058, Zhejiang, China;

3. China United Network Telecommunications Corporation Zhejiang Branch, Hangzhou 310058, Zhejiang, China;

4. Software College, Northeastern University, Shenyang 110819, Liaoning, China

Abstract: A phishing toolkit is a collection of code resources widely used for the rapid, bulk deployment of phishing websites. By identifying the specific toolkit used by a phishing website, one can efficiently detect other sites deployed using the same toolkit, supporting precise detection and attribution analysis of phishing websites. However, existing methods mostly rely on manual deployment of toolkits and suffer from low accuracy. To address these issues, this paper proposes a phishing website detection method based on phishing toolkit fingerprinting. This method extracts fingerprints of the toolkit from its directory structure, text, and resources without the need for deployment, and employs different methods to calculate fingerprint similarity. Comparing the proposed method with three baseline methods, experiments demonstrate that without deploying the toolkit, the proposed method achieves a phishing website detection accuracy of 99.5%, which is comparable to the best baseline method. Furthermore, it achieves a phishing toolkit identification accuracy

引用格式: 孙奥, 罗梦, 侯碧海, 等. 基于钓鱼工具包指纹识别的钓鱼网站检测方法[J]. 武汉大学学报(理学版), 2026, 72(4): 568-578. DOI: 10.14188/j.1671-8836.2025.0170.

SUN Ao, LUO Meng, HOU Bihai, et al. Phishing Website Detection Method Based on Phishing Toolkit Fingerprinting Identification [J]. *J. Wuhan Univ. (Nat. Sci. Ed.)*, 2026, 72(4): 568-578. DOI: 10.14188/j.1671-8836.2025.0170(Ch).

of 80.5% (an improvement of 26.2 percentage points over the best baseline) and a phishing toolkit family identification accuracy of 85.4% (an improvement of 8.6 percentage points). Ablation experiments show that the directory structure fingerprint is the core component, while text and resource fingerprints provide effective complementarity. Integrating all three features significantly improves the performance of toolkit and family identification.

Key words: phishing website; phishing toolkit; fingerprint; similarity; tree edit distance; perceptual Hash

0 引言

近年来,钓鱼网站的规模快速增长,严重威胁用户账户和隐私安全。钓鱼攻击者使用钓鱼工具包快速搭建大量钓鱼网站。

针对钓鱼网站,现有研究主要集中于检测与测量。检测方法主要基于钓鱼网站的URL^[1-3]、HTML结构^[4-5]或视觉特征^[5-8];测量工作则关注钓鱼网站的生命周期、资源依赖^[9]与躲避技术^[10]等。近年来,部分研究开始关注钓鱼工具包,通过比对钓鱼工具包与网站的资源或HTML内容^[11-13],检测网站是否使用钓鱼工具包,并识别钓鱼网站具体使用的钓鱼工具包。

通过识别钓鱼工具包检测钓鱼网站,能够批量发现同类钓鱼网站并精准溯源所用工具包。而现有钓鱼工具包识别方法通常需先部署工具包,生成可访问的钓鱼网站,并以此进行〈部署的钓鱼网站,待测网站〉的匹配。然而,部分工具包因配置复杂、缺乏关键资源,人工部署耗时甚至无法完成。此外,现有方法大多仅从文件名、资源大小等单一维度设计识别算法,识别准确率较低。

针对上述问题,本文提出一种基于钓鱼工具包指纹识别的钓鱼网站检测方法,直接提取钓鱼工具包的目录结构、文本内容与资源指纹,计算工具包和网站的指纹相似度,进行〈工具包,待测网站〉的匹配,进而准确检测使用钓鱼工具包的钓鱼网站,

并进一步识别网站具体使用的钓鱼工具包。

对比实验表明,在同样不部署工具包的前提下,本文方法在钓鱼网站检测任务上与现有方法性能相当,在钓鱼工具包(家族)识别任务中显著优于现有技术,为钓鱼网站检测和钓鱼工具包识别提供了新技术路径。

1 钓鱼工具包的多维度指纹及其表示形式

1.1 钓鱼工具包

钓鱼工具包通常开箱即用,钓鱼攻击者部署时,只需要修改数据库、邮箱等少数配置。如图1(a)所示,钓鱼工具包包含网站所需的代码、资源、配置脚本等。将该工具包部署后,可形成如图1(b)所示的钓鱼网站。

通过识别钓鱼工具包的指纹检测钓鱼网站,其核心是识别网站使用的钓鱼工具包。本文方法通过提取钓鱼工具包指纹和网站的同类指纹,计算指纹相似度以衡量钓鱼工具包和网站的相似性,进而判断网站是否为该钓鱼工具包部署形成的钓鱼网站。

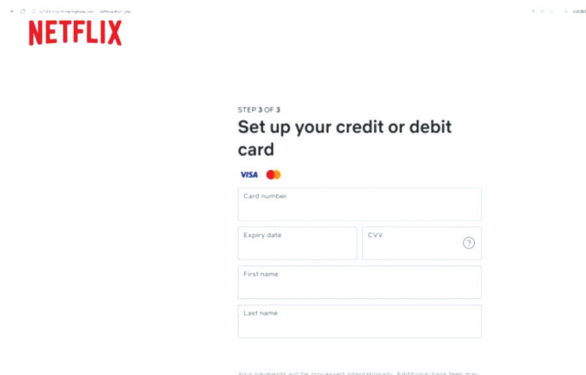
1.2 多维度指纹

钓鱼工具包指纹主要包括:目录结构指纹,文本指纹,资源指纹,视觉指纹和功能指纹。

目录结构指纹反映了钓鱼工具包文件、钓鱼网站后台文件的层级关系。钓鱼攻击者为避免因修改文件路径导致部署失败或功能异常,通常不会更



(a) 钓鱼工具包示例



(b) 钓鱼工具包部署形成的钓鱼网站示例

图1 钓鱼工具包及其部署形成的钓鱼网站

Fig. 1 Phishing toolkit and the phishing website it deploys

改代码和资源文件的目录层级与命名,目录结构较为稳定。

文本指纹反映了钓鱼工具包的文本内容,包括品牌名、诱导性语句、输入提示、联系方式等。虽然钓鱼攻击者可能修改联系方式等文本,但为保持欺骗效果,一般不会替换品牌名、网站标题等文本,文本内容具有较强的标识性。

资源指纹反映了钓鱼工具包依赖的图像、样式和脚本资源。钓鱼攻击者为保持钓鱼网站的外观与功能正常,一般不会大规模修改,网站的资源依赖较为稳定。

视觉指纹反映了钓鱼网站的外观特征,如网站的页面截图,其获取依赖网站的实际可访问性与浏览器前端渲染。

功能指纹反映了钓鱼网站的动态行为特征,如表单提交响应、页面跳转逻辑、验证码机制等,其获取同样依赖可访问网站。

相比前 3 种指纹,视觉指纹和功能指纹依赖页面渲染结果或运行时行为,需要部署工具包到服务器,过程繁琐、自动化难度高、可拓展性差,且面临部署失败的风险,不适合作为钓鱼工具包指纹。为快速、大规模部署钓鱼网站,钓鱼攻击者倾向于保持工具包的原始目录结构和文件内容,极少进行深度修改。本文通过对真实钓鱼工具包的深入观察分析,选取了钓鱼工具包的目录结构、文本、资源共三类指纹。

1.3 指纹的表示形式

本文针对目录结构、文本、资源三类指纹各自的特性,设计相应的指纹表示形式。

对于目录结构指纹,由于目录结构本质是文件、文件夹的层级关系,因此本文采用树状结构进行表示。如图 2(a)所示,树的根节点表示工具包根目录,每个子节点代表一个子目录或文件,节点名为文件名或目录名。

对于文本指纹,本文使用列表结构表示分散在标题、输入框、页脚等位置的网页文本内容,列表的每个元素为一段文本,如图 2(b)所示。

对于资源指纹,由于网站的关键资源主要包括图像、层叠样式表(Cascading Style Sheets, CSS)和 JavaScript 脚本,因此资源指纹也包括此三类,如图 2(c)所示。感知哈希(perceptual hash, pHash)面对钓鱼攻击者的裁剪、压缩、格式转换等处理时,能为图像生成近似的哈希值,因此使用 pHash 作为图像资源的指纹;CSS 资源通常不会被直接修改,因此采用 SHA-256 哈希值作为其指纹;JavaScript 资

源命名模式较为固定(如 jquery-3.2.1.slim.min.js),且文件名已较好反映其内容,因此采用文件名作为其指纹。此外,当资源文件下载失败或损坏时,使用其文件名作为替代指纹。



图 2 钓鱼工具包指纹示例
Fig. 2 Example of phishing toolkits fingerprints

1.4 逆工具包频率

不同文本和 JavaScript 资源在钓鱼工具包中出现的频率差异显著,其指纹对于钓鱼工具包识别的判别能力也不同。常见文本、通用资源的判别能力较低,而标志性文本和特有资源的判别能力更强。为量化这一判别能力,本文借鉴逆文档频率^[14],提出逆工具包频率(Inverse Kit Frequency, IKF),作为文本和 JavaScript 资源指纹的权重,如(1)式所示:

$$\text{IKF}(t) = \log \frac{N}{f} \quad (1)$$

其中,IKF 为文本或 JavaScript 资源的逆工具包频率; N 为钓鱼工具包总数; f 为该文本或 JavaScript 资源出现在钓鱼工具包中的频次。IKF 权重越高,表明文本或 JavaScript 资源越不常见。

2 本文方法

2.1 整体流程

本方法整体流程如图 3 所示,首先分别从钓鱼工具包和待测网站中提取目录结构、文本和资源指纹;随后通过不同的策略量化钓鱼工具包和网站在三类指纹上的相似度;最后取三类指纹相似度的平均值,得到工具包与待测网站的综合相似度。综合相似度的计算如(2)式所示:

$$\text{sim}_{\text{final}} = \frac{\text{sim}_{\text{dir}} + \text{sim}_{\text{text}} + \text{sim}_{\text{res}}}{3} \quad (2)$$

其中, sim_{dir} 、 sim_{text} 、 sim_{res} 为网站与工具包在目录

结构、文本和资源指纹上的相似度。

与待测网站综合相似度最高的工具包(家族), 判定为钓鱼攻击者所使用的钓鱼工具包(家族)。

当待测网站与任一工具包的综合相似度超过预设阈值 0.5 时, 判定待测网站是由钓鱼工具包部署形成的钓鱼网站。

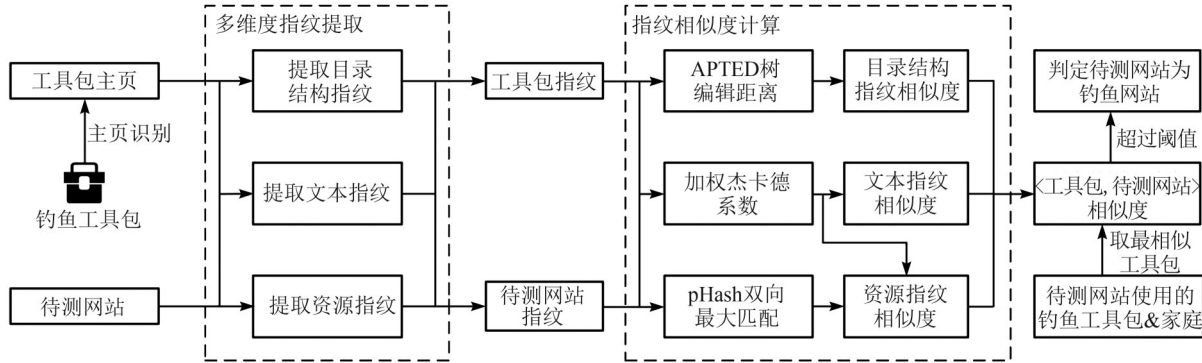


图3 基于钓鱼工具包指纹识别的钓鱼网站检测方法流程图

Fig. 3 Flowchart of the phishing website detection method based on phishing toolkit fingerprinting identification

2.2 目录结构指纹相似度

本文使用树编辑距离 (Tree Edit Distance, TED) 衡量目录结构差异。首先将工具包与网站的目录结构分别表示为树, 节点表示目录或文件。随后基于树编辑距离算法 (All Path Tree Edit Distance, APTED)^[15] 计算其最小编辑距离, 允许插入、删除、替换操作, 对应钓鱼攻击者部署时的路径修改、重命名等行为, 如图4所示。距离越小, 目录改动越小, 相似度越高。最后将其归一化为相似度, 如(3)式所示:

$$\text{sim}_{\text{dir}} = 1 - \frac{\text{TED}(T_1, T_2)}{|T_1| + |T_2|} \quad (3)$$

其中, sim_{dir} 为网站与工具包的目录结构指纹相似度; $\text{TED}(T_1, T_2)$ 为 T_1 和 T_2 两棵树的最小编辑距离; $|T_1|$ 和 $|T_2|$ 分别为两棵树的节点总数。

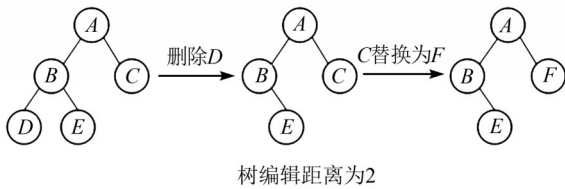


图4 树编辑距离示例

Fig. 4 Example of tree edit distance

2.3 文本指纹相似度

使用加权杰卡德 (Weighted Jaccard)^[16] 计算文本指纹相似度。设 A 和 B 分别为网站与工具包的文本集合, 则其文本指纹相似度定义如(4)式所示:

$$\text{sim}_{\text{text}} = \frac{\sum_{t \in A \cap B} \text{IKF}_{\text{text}}(t)}{\sum_{t \in A \cup B} \text{IKF}_{\text{text}}(t)} \quad (4)$$

其中, $\text{IKF}_{\text{text}}(t)$ 为文本 t 的 IKF 权重。对于已知钓

鱼工具包中未出现的文本, 取已知钓鱼工具包中文本权重的最小值, 作为其权重。

2.4 资源指纹相似度

本文将三类资源指纹相似度的算术平均, 作为最终的资源指纹相似度。如(5)式所示, 当某类资源在工具包和待测网站中均为空时 (如内嵌 CSS 代码使得无 CSS 资源引用), 此类资源不参与计算:

$$\text{sim}_{\text{res}} = \frac{\sum_{r \in R} \text{sim}_r}{|R|}, R \subset \{\text{img}, \text{css}, \text{js}\} \quad (5)$$

其中, sim_{res} 为网站与工具包的资源指纹相似度; sim_r 为网站和工具包的某类资源指纹相似度; R 为参与计算的资源种类。

2.4.1 图像资源指纹相似度

本文使用基于感知哈希 (pHash)^[17] 的双向最大匹配策略, 计算图像资源指纹相似度。如图5所示, 构建两组图像 pHash 间的相似度矩阵, 计算每行和每列最大相似值的平均值, 最终取两者平均值作为两组图像间的相似度, 如(6)式和(7)式所示:

$$\text{sim}_{\text{img}} = \frac{1}{2} \left(\frac{1}{|B|} \sum_{j=1}^{|B|} \max_{i \in A} (\text{sim}(a_i, b_j)) + \frac{1}{|A|} \sum_{i=1}^{|A|} \max_{j \in B} (\text{sim}(a_i, b_j)) \right) \quad (6)$$

$$\text{sim}(a_i, b_j) = 1 - \frac{d_H(a_i, b_j)}{n} \quad (7)$$

其中, sim_{img} 为网站与工具包之间的图像资源指纹相似度; $A = a_1, a_2, \dots, a_{|A|}$ 为网站图像资源的 pHash 集合, $B = b_1, b_2, \dots, b_{|B|}$ 为工具包图像资源的 pHash 集合; $d_H(a_i, b_j)$ 为网站图像资源和工具包图像资源两个 pHash 值之间的汉明距离; n 为 pHash 的位数 (取

64 位),用于相似度的归一化; $\max_{i \in A}(\text{sim}(a_i, b_j))$ 表示集合 A 中与 b_j 最相似的图像哈希的相似度(即相似度矩阵列方向最大值); $\max_{j \in B}(\text{sim}(a_i, b_j))$ 表示集合 B 中与 a_i 最相似的图像哈希的相似度(即相似度矩阵行方向最大值)。

列最大相似度平均值=0.855				
相似度	a_1	a_2	a_3	a_4
b_1	0.68	0.50	→ 0.95	0.81
行最大相似度 平均值=0.907	→ 0.80	0.53	0.35	0.58
b_3	0.12	↓ 0.70	0.02	→ 0.97
最终相似度 = (0.855+0.907) / 2 = 0.881				

图 5 双向最大匹配策略示例

Fig. 5 Example of the bidirectional maximum matching strategy

2.4.2 CSS资源指纹相似度

使用标准杰卡德系数计算 CSS 资源指纹相似度,如(8)式所示:

$$\text{sim}_{\text{css}} = \frac{|C \cap D|}{|C \cup D|} \quad (8)$$

其中, C 和 D 分别为网站与工具包 CSS 资源的哈希集合。

2.4.3 JavaScript资源指纹相似度

使用基于 IKF 权重的加权杰卡德系数计算 JavaScript 资源指纹相似度,突出专有脚本文件,如(9)式所示:

$$\text{sim}_{\text{js}} = \frac{\sum_{j \in E \cap F} \text{IKF}_{\text{js}}(j)}{\sum_{j \in E \cup F} \text{IKF}_{\text{js}}(j)} \quad (9)$$

其中, E 和 F 分别为网站与工具包 JavaScript 资源的文件名集合; $\text{IKF}_{\text{js}}(j)$ 为 JavaScript 资源 j 的权重。对于已知钓鱼工具包中未出现的 JavaScript 资源,取已知钓鱼工具包中 JavaScript 资源权重的最小值,作为其权重。

3 指纹提取的实现

3.1 钓鱼工具包指纹提取

本文方法无需部署钓鱼工具包,解压工具包后,通过下列步骤提取钓鱼工具包的指纹:

1) 定位工具包的有效根目录。从解压后的工具包出发,忽略“_MACOSX”等无效或空白目录,

递归进入唯一子目录,直至遇到分支或文件,此时位置即为有效根目录。

2) 确定钓鱼网站主页。遍历所有 html、php 文件,对 html 文件直接提取 HTML 内容。对 php 文件,采用标记匹配方法,识别并移除起始标记到结束标记之间的 PHP 代码块,保留剩余 HTML 内容。即,利用栈结构匹配起始与结束标记,记录所有需删除的代码区间,合并区间,最后逆序切除各区间对应的代码片段。随后解析各文件的 HTML 内容,统计表单中的输入标签数量。由于信息泄露集中在钓鱼网站主页,选取输入标签数量最多的页面作为主页。

3) 提取目录结构指纹。解析主页 HTML 中的本地资源路径,规范化路径后逐级构建嵌套字典,并纳入主页路径。如图 2(a)所示,最终每个钓鱼工具包导出 1 个嵌套字典。

4) 提取文本指纹。从主页 HTML 中提取标题、可见文本、输入提示、多行文本框提示、图片标题等,经清洗(合并空白、去首尾空格、排除单/双字符文本)后形成文本列表。如图 2(b)所示,最终每个钓鱼工具包导出 1 个文本列表。

5) 提取资源指纹。解析主页 HTML 中的资源链接,异步下载图像、CSS 和 JavaScript 文件。计算图像资源的 pHash 值、CSS 资源的 SHA-256 哈希值和 JavaScript 资源的文件名,作为资源指纹。对于下载失败或损坏的资源,使用文件名作为替代指纹。如图 2(c)所示,最终每个钓鱼工具包导出三类资源指纹的共 3 个列表。

3.2 网站指纹提取

网站的文本和资源指纹提取过程与工具包类似,区别在于输入的是网站快照中的 html 文件,而非工具包内的 php/html 文件。

目录结构指纹的提取略有不同。由于无法访问服务器真实后台,需要基于前端可见文件路径重建。以当前页面 URL 为基准,解析 HTML 中所有同站本地资源路径(如/css/style.css、img/logo.png),规范化后按层级进行拆分,并逐级构建树状嵌套字典,最后将主页路径也纳入其中。本文通过前端资源引用关系,还原网站的后台目录结构,该方法适用于仅能获取前端内容的实际检测场景。

4 实验与分析

4.1 实验环境

本实验在 Windows11 26100.4770 下进行,

CPU 型号 Intel i7-14700F, 内存大小 32 GB, 开发语言为 Python 3. 12. 7。

4.2 数据集

本实验共使用 3 个数据集, 包括钓鱼工具包数据集 (164 个工具包)、网站数据集 Tranco A (12 719 个正常网站, 164 个钓鱼网站) 和网站数据集 Tranco B (136 个正常网站, 164 个钓鱼网站)。

4.2.1 钓鱼工具包数据集

本文开发网络爬虫收集钓鱼工具包, 每 12 小时访问公开版 OpenPhish^[18]和 PhishTank^[19]钓鱼情报库, 获取并访问最新钓鱼 URL。当发现网站开启目录浏览功能 (如图 6 所示) 时, 爬虫自动遍历其后台目录并下载压缩包文件 (zip、7z、rar 格式)。对于单个钓鱼 URL, 爬虫最多访问 200 个子目录并下载 50 个压缩包作为候选。随后, 排除解压失败、加密、文件大小小于 1 KB 或不包含 htm(l)、php、py、asp(x) 等网页文件的压缩包, 获得 431 个钓鱼工具包。尽管本文方法无需部署即可检测, 但为构建已知来源的钓鱼网站样本以评估准确率, 我们随机抽取 164 个钓鱼工具包, 形成钓鱼工具包数据集, 并在后续实验中进行部署。



图6 网站的目录浏览功能

Fig. 6 Website directory listing

4.2.2 网站数据集 Tranco A

该数据集包含 12 719 个正常网站样本和 164 个钓鱼网站样本。对于正常网站部分, 本文从 Tranco Top 1M 流行网站列表^[20]中选取排名前 20 000 的网站, 使用 Selenium^[21]控制 Chrome for Testing (v134.0.6998.35) 无头浏览器^[22]访问并保存网页快照, 快照内容包括 HTML 代码、JavaScript、CSS、图片等资源。因部分网站存在反爬机制, 经人工筛选后, 最终获得 12 719 个正常网站样本。

对于钓鱼网站部分, 由于在现实网络环境中, 难以确认钓鱼网站具体使用的钓鱼工具包, 因此本文部署钓鱼工具包数据集中的工具包, 修改表单提交地址等配置信息, 确保部署形成的网站外观和功能正常 (如图 1(b) 所示)。本文成功在内网服务器 (Apache 2.4.39+PHP 7.3.4) 部署 164 个工具包,

随后使用 Selenium 采集其网页快照, 人工排除无效样本, 最终得到 164 个带有真实工具包标签的钓鱼网站样本。

4.2.3 网站数据集 Tranco B

该数据集包含 136 个正常网站样本和 164 个钓鱼网站样本。本文从 Tranco Top 1M 列表中选取前 200 的网站, 采用与数据集 A 相同的爬取和筛选流程, 最终获得 136 个正常网站样本。钓鱼网站部分则与数据集 Tranco A 完全一致, 复用其 164 个已标注的钓鱼网站样本。

4.3 评价指标

为评估本方法在钓鱼网站检测和钓鱼工具包 (家族) 识别任务中的性能, 本实验采用 3 种评价指标: 钓鱼网站检测准确率、钓鱼工具包识别准确率、钓鱼工具包家族识别准确率。

钓鱼网站检测: 给定测试正常网站或钓鱼网站, 若正确判断网站是否使用了钓鱼工具包, 则视为检测正确。

钓鱼工具包识别: 给定测试钓鱼网站, 若相似度最高的钓鱼工具包与网站实际使用的钓鱼工具包一致, 则视为正确识别。该指标衡量钓鱼工具包精确识别能力。

钓鱼工具包家族识别: 给定测试钓鱼网站, 若相似度最高的钓鱼工具包所属的家族与钓鱼网站实际使用的钓鱼工具包所属的家族一致, 则视为识别正确。该指标衡量识别方法在面对工具包变种等现实挑战时的鲁棒识别能力。

在钓鱼工具包家族识别准确率评估中, 钓鱼工具包的家族由人工分析确定。本文使用 diff 工具^[23]对比工具包代码差异, 结合钓鱼工具包主页的外观与功能差异, 从 164 个钓鱼工具包中形成 70 个钓鱼工具包家族, 其中 45 个家族仅有一个钓鱼工具包, 具体如表 1 所示。

4.4 常见的文本与资源

本文方法在计算文本指纹相似度和 JavaScript 资源指纹相似度时, 使用逆工具包频率度量文本和资源的出现频率, 权重越高说明其出现频率越低, 适合作为标志性指纹。在参与实验的 164 个钓鱼工具包中, 最常见和最不常见的 JavaScript 资源如表 2, 最常见和最不常见的文本如表 3。

4.5 对比实验

本文从钓鱼工具包识别相关工作中, 选取 3 个具有代表性的基线方法进行复现, 并在钓鱼工具包数据集和网站数据集 Tranco A 上, 与本方法进行对比实验。本文选取的基线方法来自文献[11-13], 3


```
"ffreedx.serv00.net_dhl": [  
  "css-541951", "js-7442",  
  "img-1150", "img-1603",  
  "js-18430", "img-2054",  
  "img-11968", "js-86841",  
  "img-40819", "js-301907",  
  "img-865"]
```

图9 文献[13]方法的指纹示例

Fig. 9 Example of fingerprint for Ref. [13]

换为直接从原始工具包中提取的对应内容(如HTML结构、资源)。而文献[11]本身即支持直接从工具包提取指纹,无需部署,因此无需替换。

在复现过程中,我们遇到实现细节缺失的问题:文献[11]未公开wFuzz使用的爆破字典,导致“潜在资源”指纹提取不完整;文献[12]未指定HTML序列相似度的具体算法,本实验采用主流HTML结构相似度工具^[24]替代;文献[13]中的“通用资源过滤”依赖其未开源的网站访问日志统计结果,本实验省略该预处理步骤。尽管存在上述调整,但所保留的核心指纹提取逻辑仍能反映各方法的本质能力。

表4展示了本文方法与3种基线方法的对比结果。

表4 对比实验结果

Table 4 Contrast experiment results

方法	钓鱼网站	钓鱼工具包	钓鱼工具包家族
	检测准确率	识别准确率	识别准确率
文献[11]	9.0	48.8	61.0
文献[12]	99.4	52.4	73.2
文献[13]	99.5	54.3	76.8
本文方法	99.5	80.5	85.4

在钓鱼网站检测准确率方面,本文方法达到99.5%,与最佳基线方法文献[13]相当。而文献[11]由于相似度阈值设置过低,将大量正常网站误判为使用了钓鱼工具包的钓鱼网站,钓鱼网站检测准确率显著低于本方法。

在钓鱼工具包识别准确率方面,本方法达到80.5%,较最佳基线方法文献[13]提升了约26.2个百分点。这是因为文献[13]基于<资源类型,资源大小>的重合程度进行识别,虽在一定程度上反映了资源依赖关系,但由于忽略了资源文件名语义与网页内容,准确率受限(54.3%)。文献[11]仅通过资源路径与文本的重合程度进行识别,因此准确率最低(48.8%)。文献[12]依赖HTML标签序列的结构相似性,虽对页面布局变化具有一定鲁棒性,但易受标签顺序扰动影响,性能有限(52.4%)。

本文方法的钓鱼工具包家族识别准确率为85.4%,较最佳基线方法文献[13]提升了约8.6个百

分点。这一提升主要得益于本文对多维度指纹的融合:目录结构指纹捕捉了文件层级关系的稳定性,文本指纹通过加权机制增强了对标志性语句的判别能力,资源指纹综合了图像视觉语义、CSS样式语义与JavaScript功能语义。此外,本文提出的逆工具包频率(IKF)降低了常见资源的干扰。

综上所述,在不部署钓鱼工具包的前提下,本方法具备精准的钓鱼网站检测能力,且在钓鱼工具包识别任务中表现优异,并在反映工具包修改等现实情况的钓鱼工具包家族识别任务上,仍具备较强的识别能力。

4.6 消融实验

由于网站数据集Tranco A中正常网站比例较高,不同指纹组合之间的钓鱼网站检测准确率都处于较高水平,差异很小,为更直观地展现不同指纹组合的差异,评估各类指纹的贡献,本文在钓鱼工具包数据集和网站数据集Tranco B上,开展消融实验。具体实验结果如表5所示,其中双指纹组合每类指纹的权重设为0.5。

在钓鱼网站检测任务上,本方法仅使用目录结构或文本指纹时,钓鱼网站检测准确率均达到79.7%,为各指纹组合中的最高值,而融合全部三类指纹后,检测准确率略微下降至78.0%,仍与最优基线方法相当(文献[13],77.3%)。这表明指纹信息的融合引入了噪声。而在钓鱼工具包(家族)识别任务上,三类指纹的融合取得显著效果:

1) 单指纹组合中,目录结构指纹的钓鱼工具包识别准确率(65.9%)显著优于最佳基线方法(文献[13],54.3%),表明其有效性。相比之下,文本(56.7%)和资源指纹(46.3%)在单独使用时判别能力有限,与基线方法相当。

2) 双指纹组合中,文本与资源融合的提升有限,表明两者存在信息重叠。但两者分别与目录结构结合时均取得显著提升。“目录结构+文本”组合在工具包识别上更强(78.0%),而“目录结构+资源”组合在工具包家族识别上更优(83.5%),表明资源比文本更能反映工具包家族特征。

3) 三指纹组合在工具包识别(80.5%)和工具包家族识别(85.4%)上均达到最佳性能,表明文本与资源能独立且互补地增强目录结构的判别能力,验证了多维度指纹融合的有效性。

4.7 误报与漏报分析

在对比实验中,本文在钓鱼工具包数据集与Tranco A网站数据集(12 719正常网站,164个钓鱼网站)评估了所提方法。结果显示,该方法在钓鱼

表 5 消融实验结果

Table 5 Ablation experiment results

指纹组合/方法	钓鱼网站	钓鱼工具包	钓鱼工具包家族
	检测准确率	识别准确率	识别准确率
文献[11]	56.0	48.8	61.0
文献[12]	76.0	52.4	73.2
文献[13]	77.3	54.3	76.8
目录结构	79.7	65.9	68.3
文本	79.7	56.7	73.8
资源	69.7	46.3	68.9
文本+资源	78.3	57.3	75.6
目录结构+资源	72.3	76.8	83.5
目录结构+文本	73.3	78.0	82.3
目录结构+文本+资源	78.0	80.5	85.4

网站检测任务中实现了 0% 的误报率,但漏报率达 40.2%。对漏报进行分析,发现漏报产生的主要原因有以下三类:

1) JavaScript 动态加载:如图 10(a)所示,部分网站依赖 JavaScript 动态生成内容,而本方法采用静态 HTML 解析,导致提取的文本和资源指纹,与页面实际渲染结果存在偏差,引发漏报。

2) 钓鱼工具包混淆:如图 10(b)所示,部分基于 PHP 的钓鱼工具包使用代码混淆技术,导致本方法从 php 文件中提取 HTML 内容时出现残缺或失败,导致有效指纹缺失,引发漏报。

3) 指纹稀疏:如图 10(c)所示,部分工具包设计简单、静态资源少、目录结构扁平,可提取的目录与资源指纹不足,指纹信息的匮乏加剧了漏报风险。

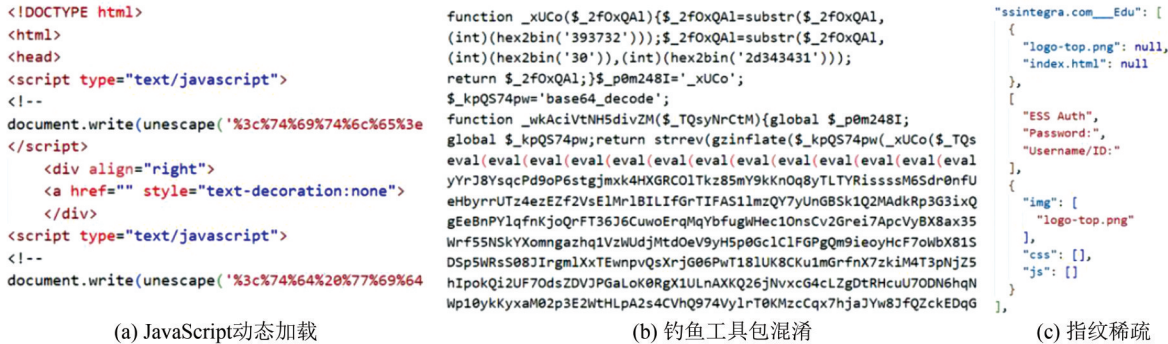


图 10 漏报原因

Fig. 10 Reasons for false negatives

5 讨论

本研究提出的基于钓鱼工具包指纹识别的钓鱼网站检测方法,在不依赖工具包部署的情况下,有效提升了钓鱼工具包及其家族的识别准确率。然而本文方法也还存在一些不足,有待在下一步研究中改进、完善。

1) 本文中,钓鱼工具包家族主要依据 diff 工具的代码比对结果,以及主页外观的相似性,人工进行划分。该方式具有一定合理性,但依赖人工判断,未来需构建明确、量化的划分标准。例如基

于代码相似性的聚类分析,利用代码指纹、抽象语法树编辑距离或代码嵌入技术计算工具包间的相似度,并通过设定阈值或应用聚类算法实现家族划分。此外,还可结合资源依赖等特征,构建更客观的划分标准。

2) 本文方法面对 JavaScript 动态加载、工具包混淆、极简工具包等问题时仍存在局限,可能导致指纹提取不完整,进而引发误报和漏报。对此,可通过以下技术进行改进:① 模拟浏览器环境,通过执行 JavaScript 获取完全渲染的 DOM,以提取完整指纹;② 利用代码反混淆技术,还原经混淆的代码

逻辑,确保指纹的正常提取;③利用深度学习技术,学习高阶隐含特征,提升本文方法在极简工具包特征稀疏情况下的检测能力。

3)本研究在复现基线方法时也面临一定限制。由于部分文献未开源关键实现细节(如文献[11]中wFuzz的爆破字典、文献[13]中通用资源的统计过滤规则),我们仅能复现其核心指纹提取逻辑,而省略了依赖外部数据的辅助模块。尽管这些模块并非识别算法的关键部分,但仍可能对最终性能产生影响。未来工作可推动社区建立标准化的钓鱼工具包基准测试框架,统一输入格式与评估方法,以提升方法间对比的公平性与可复现性。

6 结 语

为准确检测使用钓鱼工具包部署的钓鱼网站,针对现有方法大多依赖人工部署、识别维度单一且准确率有限的问题,本文提出一种基于钓鱼工具包指纹识别的钓鱼网站检测方法,基于对钓鱼工具包“开箱即用、极少修改”特性的观察,构建了目录结构、文本与资源三类稳定指纹,并设计了差异化的相似度计算策略,实现了对钓鱼网站及其所用工具包的精准检测。

对比实验结果表明,在不部署钓鱼工具包的前提下,本方法在钓鱼网站检测任务上与文献[13]基线方法相当。而在钓鱼工具包识别和钓鱼工具包家族识别任务上,本方法显著优于基线方法,较文献[13]基线方法分别提高26.2个百分点和8.6个百分点。消融实验结果表明,多维度指纹的融合显著提升了钓鱼工具包识别和钓鱼工具包家族识别能力。

未来工作可结合代码相似性和多模态特征,构建可量化的钓鱼工具包家族划分标准;结合反混淆技术和程序动态分析技术,强化方法对动态网页、混淆代码与极简工具包的鲁棒性;建立标准化的钓鱼工具包基准测试框架,提升研究的可复现性与方法对比的公平性。

参考文献:

- [1] BOZKIR A S, DALGIC F C, AYDOS M. GramBeddings: A new neural network for URL based identification of phishing web pages through N -gram embeddings[J]. *Computers & Security*, 2023, **124**: 102964. DOI: 10.1016/j.cose.2022.102964.
- [2] ZHENG F A, YAN Q, LEUNG V C M, *et al.* HDP-CNN: Highway deep pyramid convolution neural network combining word-level and character-level representations for phishing website detection[J]. *Computers & Security*, 2022, **114**: 102584. DOI: 10.1016/j.cose.2021.102584.
- [3] 钟文康, 王添, 张功萱. 基于组件分割的钓鱼URL检测方法[J]. *信息安全学报*, 2025, **10**(1): 130-142. DOI: 10.19363/J.cnki.cn10-1380/tn.2025.01.10.
ZHONG W K, WANG T, ZHANG G X. Phishing URL detection method based on component segmentation[J]. *Journal of Cyber Security*, 2025, **10**(1): 130-142. DOI: 10.19363/J.cnki.cn10-1380/tn.2025.01.10 (Ch).
- [4] PURWANTO R W, PAL A, BLAIR A, *et al.* Phish-Sim: Aiding phishing website detection with a feature-free tool[J]. *IEEE Transactions on Information Forensics and Security*, 2022, **17**: 1497-1512. DOI: 10.1109/TIFS.2022.3164212.
- [5] 张思睿, 延志伟, 董科军, 等. 基于注意力机制与特征融合的网络钓鱼检测算法[J]. *网络与信息安全学报*, 2024, **10**(4): 123-131. DOI: 10.11959/j.issn.2096-109x.2024058.
ZHANG S R, YAN Z W, DONG K J, *et al.* Phishing detection algorithm based on attention and feature fusion[J]. *Chinese Journal of Network and Information Security*, 2024, **10**(4): 123-131. DOI: 10.11959/j.issn.2096-109x.2024058 (Ch).
- [6] LIU R F, LIN Y, YANG X D, *et al.* Inferring phishing intention *via* webpage appearance and dynamics: A deep vision based approach[DB/OL]. [2025-08-02]. https://www.usenix.org/system/files/sec22fall_liu-ruofan.pdf.
- [7] LIN Y, LIU R F, DIVAKARAN D M, *et al.* Phishpedia: A hybrid deep learning based approach to visually identify phishing webpages [DB/OL]. [2025-08-02]. <https://www.usenix.org/system/files/sec21fall-lin.pdf>.
- [8] 文伟平, 朱一帆, 吕子晗, 等. 针对品牌的网络钓鱼扩线与检测方案[J]. *信息安全*, 2023, **23**(12): 1-9.
WEN W P, ZHU Y F, LÜ Z H, *et al.* Brand-specific phishing expansion and detection solutions [J]. *Netinfo Security*, 2023, **23**(12): 1-9 (Ch).
- [9] LIM K, PARK J, KIM D. Phishing *vs.* legit: Comparative analysis of client-side resources of phishing and target brand websites[C]//Proceedings of the ACM Web Conference 2024. New York: ACM, 2024: 1756-1767. DOI: 10.1145/3589334.3645535.
- [10] ZHANG P H, OEST A, CHO H, *et al.* CrawlPhish: Large-scale analysis of client-side cloaking techniques in phishing[C]//2021 IEEE Symposium on Security and Privacy (SP). New York: IEEE Press, 2021: 1109-1124. DOI: 10.1109/SP40001.2021.00021.
- [11] BIJMAN H L J, BOOI T M, SCHWEDERSKY A, *et al.* Catching phishers by their bait: Investigating the

- Dutch phishing landscape through phishing kit detection [C]// In 30th USENIX Security Symposium (USENIX Security 21). Berkeley: USENIX Association, 2021: 3757-3774.
- [12] LEE W, HUR J, KIM D. Beneath the phishing scripts: A script-level analysis of phishing kits and their impact on real-world phishing websites [C]//Proceedings of the 19th ACM Asia Conference on Computer and Communications Security. New York: ACM, 2024: 856-872. DOI:10.1145/3634737.3657013.
- [13] TANAKA S, MATSUNAKA T, YAMADA A, *et al.* Phishing site detection using similarity of website structure [C]//2021 IEEE Conference on Dependable and Secure Computing (DSC). New York: IEEE Press, 2021: 1-8. DOI:10.1109/DSC49826.2021.9346256.
- [14] SPARCK JONES K. A statistical interpretation of term specificity and its application in retrieval[J]. *Journal of Documentation*, 1972, **28** (1): 11-21. DOI: 10.1108/eb026526.
- [15] PAWLIK M, AUGSTEN N. Efficient computation of the tree edit distance[J]. *ACM Transactions on Database Systems*, 2015, **40**(1): 1-40. DOI:10.1145/2699485.
- [16] JACCARD P. The distribution of the flora in the Alpine zone[J]. *The New Phytologist*, 1912, **11**(2): 37-50.
- [17] JOHANNES B. ImageHash 4.3.2[EB/OL]. [2025-02-01]. <https://pypi.org/project/ImageHash/>.
- [18] OpenPhish. OpenPhish Phishing Intelligence [EB/OL]. [2025-08-15]. <https://openphish.com/>.
- [19] Cisco Talos Intelligence Group. PhishTank [EB/OL]. [2025-08-15]. <https://phishtank.org/>.
- [20] VICTOR L P. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation [EB/OL]. [2025-08-15]. <https://tranco-list.eu/>.
- [21] JASON H. Selenium 4.35 [EB/OL]. [2025-08-12]. <https://www.selenium.dev/>.
- [22] Google. Chrome for Testing [EB/OL]. [2025-08-15]. <https://developer.chrome.google.cn/blog/chrome-for-testing-hl=zh-cn>.
- [23] Free Software Foundation. GNU Diffutils [EB/OL]. [2025-08-15]. <https://www.gnu.org/software/diffutils/>.
- [24] MATISKAY. Html-similarity[EB/OL]. [2020-09-22]. <https://github.com/matiskay/html-similarity>.

□