密码学报 (中英文) ISSN 2097-4116 CN 10-1900/TN
Journal of Cryptologic Research, 2024, 11(1): 67–100

E-mail: jcr@cacrnet.org.cn
http://www.jcr.cacrnet.org.cn
Tel/Fax: +86-10-82789618

创刊十周年纪念特刊

口令猜测研究进展*

邹云开[1,2,3], 汪 定[1,2,3]

1. 南开大学 网络空间安全学院, 天津 300350
2. 天津市网络与数据安全技术重点实验室, 天津 300350
3. 数据与智能系统安全教育部重点实验室, 天津 300350

通信作者: 汪定, E-mail: wangding@nankai.edu.cn

摘 要: 口令是人类可记忆的短密钥, 在身份认证、加密、签名等领域有广泛的应用. 口令虽然被指出存在一系列安全性和可用性问题, 但因其使用简单、成本低廉、容易更改, 在可预见的未来仍无可替代. 口令猜测是口令面临的最严重的安全威胁, 是口令安全性研究的核心方向之一, 引起了学术界的持续关注. 本文首先采用数据驱动的研究方法, 挖掘可被猜测攻击者利用的用户脆弱口令行为, 分析用户口令构造规律, 包括流行特性、语言依赖性、长度分布、口令重用、结构和语义特征等方面. 接着, 总结了近 30 年来学术界提出的 28 种主要口令猜测算法, 并根据技术原理的不同对其进行分类. 随后, 回顾了目前广泛使用的口令猜测算法评估指标, 探究了不同实验设置对评估算法攻破率和计算效率的影响, 并根据实验结果讨论了不同猜测算法的技术特点和适用场景. 最后, 总结口令猜测领域的研究进展, 并展望口令猜测算法的应用领域和未来的研究方向.

关键词: 密钥安全; 口令; 口令强度; 口令猜测; 机器学习

中图分类号: TP309.7 **文献标识码:** A DOI: 10.13868/j.cnki.jcr.000670

中文引用格式: 邹云开, 汪定. 口令猜测研究进展[J]. 密码学报 (中英文), 2024, 11(1): 67–100. [DOI: 10.13868/j.cnki.jcr.000670]

英文引用格式: ZOU Y K, WANG D. Advances on password guessing attack[J]. Journal of Cryptologic Research, 2024, 11(1): 67–100. [DOI: 10.13868/j.cnki.jcr.000670]

Advances on Password Guessing Attack

ZOU Yun-Kai[1,2,3], WANG Ding[1,2,3]

1. College of Cyber Science, Nankai University, Tianjin 300350, China
2. Tianjin Key Laboratory of Network and Data Security Technology, Tianjin 300350, China
3. Key Laboratory of Data and Intelligent System Security, Ministry of Education, Tianjin 300350, China

Corresponding author: WANG Ding, E-mail: wangding@nankai.edu.cn

Abstract: Passwords are usually short, memorable keys used in various applications such as identity authentication, encryption, and digital signature. While some security and usability issues of passwords

* 基金项目: 国家自然科学基金 (62172240, 62222208)
Foundation: National Natural Science Foundation of China (62172240, 62222208)
收稿日期: 2023-10-08 定稿日期: 2024-02-19

have been identified, the simplicity, cost-effectiveness, and ease of change make passwords irreplaceable in the foreseeable future. Password guessing poses the most serious security threat to passwords, serving as a central focus in password security research and garnering sustained attention from the academic community. This paper employs a data-driven approach to unearth user behaviors that can be exploited by potential attackers in guessing passwords. It analyzes patterns in password creation, encompassing aspects like popularity trends, language dependencies, length distributions, password reuse, structural and semantic features. Subsequently, this paper summarizes 28 mainstream password guessing algorithms proposed over the past three decades, classifies them based on their technical principles. Following that, this paper reviews the widely used evaluation metrics for password guessing algorithms, explores the impact of different experimental setups on algorithm performance, and discusses the technical characteristics and application scenarios of various guessing algorithms based on experimental results. Finally, this paper presents a comprehensive overview of the research advancements in password guessing and offers insights into practical applications and future research directions in the field.

Key words: secret key security; passwords; password strength; password guessing; machine learning

1 引言

口令, 英文为 “Password”, 坊间称为 “密码”. 口令是人类可记忆的短密钥, 是全球 53 亿网民可感知、直接接触的密码学组件. 口令最早由麻省理工学院的费尔南多·科尔巴托教授引入计算机领域, 用于大型计算机的本地文件访问控制 [1]. 上世纪 90 年代, 随着个人计算机和互联网的迅速发展, 各类服务不断上网, 口令除了提供数据加密、数字签名等安全功能, 也成为使用最广泛的身份认证方式, 保护着用户绝大多数的数字资产. 然而, 随着用户需要管理的网络账户数量的增加, 用户在创建口令时也面临着巨大的安全挑战. 一方面, 人类的记忆力有限且长期保持稳定, 通常只能记忆 5∼7 个不同口令 [2]; 另一方面, 许多网站对口令的复杂度和长度提出了明确的要求 (如网页托管服务公司 000Webhost 要求用户创建的口令长度至少为 6, 且至少包含一个数字和字母), 使得用户记忆众多账户的口令变得愈发困难. 这导致许多用户为了方便记忆而采用简单易记的流行口令, 在不同账户中使用相同口令, 使用个人信息创建口令 [3]. 这些脆弱行为一旦被攻击者利用, 将导致严重的安全隐患.

自 2000 年以来, 数以千计的各种各样新型身份认证方案被提出, 如多因素认证 [4]、生物识别技术 [5] 以及硬件安全密钥 [6] 等, 但这些方案在可用性、安全性和隐私保护等方面往往没有取得平衡 [7,8]. 同时, 它们在可部署性上也常常劣于传统的口令认证 [9]. 因此, 自 2012 年以来, 学术界逐渐达成共识 [8,10–12]: 在可预见的未来, 口令仍将是最主要的身份认证方法, 基于口令的认证技术仍无可替代.

随着信息化进程的不断推进, 口令所面临的 “可记忆与抗猜测” 要求之间的矛盾愈发凸显. “可记忆” 要求口令尽量短、有规律、不复杂, “抗猜测” 要求口令尽量长、无规律、越复杂越好. 令人更为担忧的是, 近年来频繁发生的大规模口令泄露事件 (如 32 亿条包含邮箱和口令的 COMB 数据合集 [13]), 为攻击者猜测用户口令提供源源不断的素材. 同时攻击者的能力也在不断增强, 如深度学习技术的进展为提高口令猜测效率提供了潜在的新途径 [14–16] . 鉴于口令往往作为保护信息系统的第一道防线, 从攻击者的视角出发, 深入研究口令在猜测攻击下的安全强度变得尤为重要.

口令猜测方法根据技术路线的不同, 主要可分为三类. 第一类为基于规则的猜测攻击, 如将口令 `password` 通过首字母大写、字母 `a` 跳变为字符 `@` 的规则, 转化为 `P@ssword`. 此类方法最早可追溯到 1979 年, Morris 和 Thompson [17] 首次设计了一系列启发式变换规则, 以生成字典中单词的变体, 用于口令猜测攻击. 随后, 一些口令破解工具 (如 John the Ripper [18] 和 Hashcat [19]) 通过利用 GPU 来进行大规模自动化的猜测攻击, 进一步推动了基于规则的口令猜测技术的发展. 时至今日, 这些工具仍是现实攻击者进行离线口令猜测攻击的主要手段之一.

第二类为基于统计学方法的猜测攻击. 这类方法主要通过统计训练口令字典中基本元素的频次来设计概率模型, 典型的代表有马尔科夫链 (Markov [20]) 和概率上下文无关文法 (PCFG [21]). 早在 2005 年,

Narayanan 和 Shmatikov [20] 提出了一种基于马尔科夫链的口令猜测模型. 该模型通过估算相邻两个字符之间的概率, 揭示了利用自然语言处理 (NLP) 技术来生成口令猜测的潜力. 在此基础上, Ma 等人 [22] 于 2014 年在马尔科夫链上引入了多种归一化和平滑技术, 以克服原始模型存在的数据稀疏和过拟合问题. 此外, 在 2009 年, Weir 等人 [21] 提出了基于概率上下文无关文法的口令猜测模型 PCFG. 该模型将口令视为多个独立的、不同类型的字符串组成 (如字母段 L、数字段 D 和特殊字符段 S), 利用基于统计数据获得的文法规则生成口令结构模板, 随后用训练字典中的字符串填充这些模板. 此后, 大量基于 PCFG 的猜测方法被陆续提出. 例如, 在 2014 年, Veras 等人 [23] 进一步对字母 L 段进行语义挖掘, 提出了融合语义信息的 Semantic-PCFG 算法.

第三类为基于机器学习/深度学习技术的猜测攻击. 这类方法通常将口令猜测视为文本生成任务, 利用现有的机器学习模型学习口令的结构与语义特征. 训练完成后, 模型可自动生成大规模猜测. 2016 年, Melicher 等人 [14] 首次将深度学习技术引入到口令猜测领域, 并提出了基于循环神经网络的口令猜测模型 FLA (fast, lean, and accurate). 与 Markov 模型类似, FLA 也通过 n 阶字符串预测下一个字符, 不同之处在于, FLA 在训练完成后, 自动为字符表中的每个字符都分配一个较小的 (非零的) 概率, 作为输入 n 阶字符串的预测结果. 该方法通常在较大猜测规模 (如 $>10^{10}$ 猜测数) 开始展现优势. 近年来, 随着生成式对抗网络 (GAN) 技术在自然语言处理 (NLP) 领域的发展, 涌现出一批基于 GAN 的口令猜测模型, 如 PassGAN [15] 和条件性口令猜测 (conditional password guessing, CPG) [16]. 此外, 还有研究将深度学习技术与传统的基于规则或基于统计学的口令猜测方法相结合, 进一步提升了现有猜测方法的攻击效率. 例如, 2021 年, Pasquini 等人 [24] 提出了自适应动态规则调整攻击方法 AdaMs (adaptive, dynamic mangling rules). 该方法利用深度学习技术优化了口令与对应规则列表的匹配优先级, 显著提升了 (基于规则的) 字典攻击的攻击效率.

在研究口令猜测算法时, 如何准确且公平地评估其优劣是一个关键问题. 当前, 学术界主要采用 “猜测成功率 vs. 猜测数” 这一指标来评价不同算法的优劣, 即给定猜测数下, 算法能成功命中口令的比例越高, 算法越好. 由于该指标与攻击者所拥有的计算资源和运行算法所需要的系统环境配置无关 [25], 只取决于用户口令强度和算法自身的猜测能力, 因此, 这一评价标准逐渐成为公平对比各个口令猜测算法的重要手段. 然而, 在现实世界中, 计算资源与环境配置往往是确定的, 在此情况下, 不同类型算法的计算效率差异巨大. 例如, 在普通服务器上 (CPU: Xeon Silver 4210R 2.4 GHz; GPU: GeForce RTX 3090), 基于统计学方法的 PCFG [21] 每秒可生成 8 万猜测, 而基于机器学习技术的 RFGuess [26] 生成同样数量的猜测需要约 10 分钟. 虽然更换更好的硬件设备能提升对应算法的计算效率, 但这涉及到巨大的攻击成本差异. 因此, 计算效率和攻击成本理应成为猜测算法的评价指标之一.

此外, 科学合理的实验设置也是评估猜测算法的重要因素. 尽管当前关于口令猜测的研究众多, 但在实验设置的细节方面存在较大差异, 其中一些关键设置至今仍未形成普遍共识. 在此有以下几个关键方面需要特别关注: 首先是数据集比例的划分方式. 不同研究采用了不同的训练集和测试集比例, 如 80% 对 20% [15,16]、50% 对 50% [21,27,28] 甚至 75% 对 25% [26]. 与之紧密相关的是训练集和测试集的大小问题. 以 Veras 等人 [29] 的研究为例, 他们发现 PCFG 算法在超过百万量级的训练集上, 进一步增加训练数据的规模并未显著提高猜测成功率. 另一方面, Dell'Amico 等人 [30] 在对不同算法采用蒙特卡洛方法进行评估时, 仅仅选用了 1 万大小的测试集, 而 Pal 等人 [31] 的实验结果表明, 当测试集达到 10 万时, 基于口令重用的定向猜测算法的破解成功率趋于稳定 (误差小于 0.1%). 其次, 训练集和测试集的选取问题也具有重要意义. 在现实场景中, 攻击者往往是聪明的, 因此, 所使用的训练集和测试集的口令分布应越接近越好, 即训练集与测试集在语言、服务类型和口令策略上应尽可能保持一致. 然而, 当前的研究, 包括一些最新的工作 (如文献 [16,32,33]), 在实验设置时并未明确考虑这些关键细节.

进一步, 还需要考虑训练集和测试集之间的交叉问题. 一些研究 (如文献 [15,21,33]) 在测试集中去除了训练集中已出现过的口令, 以评估算法生成新口令的能力, 即泛化能力. 然而, 与之相对应的是, Wang 等人 [26] 的研究指出, 攻击者在现实场景中往往无法了解被攻击目标的口令分布, 因此更建议保留交叉部分, 避免去除测试集中与训练集重合的部分. 此外, 生成训练集中已存在口令的能力, 即重新排列训练集中的部分口令, 也能够反映猜测算法的拟合能力.

面对以上关键问题, 需要深入探讨: 不同比例的训练集与测试集划分是否对猜测算法评估结果产生显

著影响？使用何种规模的训练集与测试集是科学合理的？是否有必要保持训练集与测试集的分布尽可能一致？是否应该去除训练集与测试集的重合/交叉部分？回答这些问题将有助于确立更科学的实验设置原则, 以便更准确地评估口令猜测算法的性能 (包括攻破率和计算效率).

目前, 针对口令猜测的研究主要聚焦于算法本身, 缺乏对算法在不同场景下实际应用效果的系统讨论. 一些基于机器学习的猜测算法 (如 FLA [14]), 受限于口令生成速度, 更适合作为 PSM (password strength meter) 来实际应用. 虽然基于统计学的猜测算法 (如 PCFG [21]) 生成速度较快, 但其往往受限于训练集, 在大猜测数下的表现出现瓶颈. 此外, 在现实情况下, 攻击者可能采取多种手段来进行口令猜测, 如何在给定计算资源的条件下, 选取更高效的猜测算法是一个值得深入探讨的问题.

另一个值得关注的方向是口令文件泄露检测, 即利用猜测算法生成诱饵口令 (decoy passwords, 即 honeywords [27,34]), 并与用户真实口令混合存储, 通过监测诱饵口令的登录尝试来判断口令文件是否泄露. 然而, 学术界提出的各类口令强度评价算法 (特指基于猜测算法的 PSM) 和诱饵口令机制, 均未在实际系统中得到广泛部署, 阻碍它们实际应用的障碍与挑战也是一个值得重点关注的议题.

无论是面向国家安全还是个人用户的数字资产保护, 对口令猜测技术进行全面深入研究都具有迫切现实需求. 近年来, 口令猜测研究逐渐成为一个热点, 涌现出了一大批关于口令猜测的研究成果 (如文献 [16, 24, 26, 32, 33, 35]). 2023 年, Yu 等人 [36] 回顾了 2016–2023 年学术界提出的 37 种口令猜测算法, 并展示了口令猜测领域相关研究的交叉引用趋势. 然而, 该研究主要关注基于神经网络的口令猜测算法, 并未对不同口令猜测算法之间的性能差异进行深入探讨, 缺乏对已有算法优劣的系统总结. 为此, 本文对口令猜测领域的已有主流算法进行了全面的梳理和分类, 并指出该领域存在的研究局限性和未来值得研究的方向. 这将有助于后续的研究者审视最新方法, 并避免重复研究.

综上, 本文将首先基于大规模真实口令数据, 分析用户口令构造规律, 挖掘用户的脆弱口令行为, 再围绕不同类型的口令猜测算法, 从技术原理、实验设置 (包括算法评估指标)、实际应用与未来研究方向 4 个方面, 对国内外最新口令猜测研究进展进行综述.

2 用户口令规律分析

自 2009 年以来, 大规模口令数据泄露事件频繁发生. 例如, 2009 年, 社交媒体应用开发商 Rockyou 遭遇黑客攻击, 导致其 3200 万用户的口令以明文形式被泄露 [37]; 2011 年, 国内多家知名互联网公司 (如 CSDN、天涯、人人网) 发生大规模用户口令数据泄露 [38]; 2012 年, 知名云存储服务商 Dropbox 遭遇数据泄露, 6800 万个账户信息被黑客窃取 [39]; 2014 年, 国内知名购票软件 12306 发生数据泄露, 泄露信息包含用户账号、明文口令、身份证和邮箱等多种个人信息 [40]; 2015 年, 知名的游戏直播平台 Twitch 遭遇数据泄露事件, 导致约 5500 万用户口令文件被泄露 [41]; 2016 年, Yahoo 公司宣称其数十亿用户信息在 2013 年被泄露, 包括姓名、邮箱和口令等 [42]; 2017 年, DNA 检测公司 MyHeritage 遭遇黑客入侵, 约 9300 万用户的电子邮箱和口令被泄露 [43]; 2018 年, 华住集团发生数据泄露, 包括 1.3 亿条身份信息、2.4 亿条开房记录等, 共计约 5 亿条信息 [44]; 2019 年, 1.62 亿 Dubsmash (多媒体应用) 用户数据被泄露, 包括姓名、用户名和口令等 [45]; 2020 年, 知名视频会议软件 zoom 的 53 万用户口令被泄露 [46]; 2021 年, Nitro PDF 数据库超 7700 万条数据被泄露, 包括邮箱、用户名和口令等 [47]; 2022 年, 黑客入侵了著名的宠物游戏网站 Neopets 的数据库, 并窃取了 6900 万用户的个人数据, 包括用户名、邮箱和口令等 [48]. 这些源源不断泄露的口令数据为口令猜测研究提供了有力的数据支撑.

本文首先将从流行特性、语言依赖性、长度分布、口令重用、结构和语义特征 (如包含个人信息) 等 6 个方面挖掘用户的脆弱口令行为, 分析用户口令构造规律.

2.1 数据集简介

表 1 展示了本文所采用的口令数据集, 总共包括 6 个中文和 7 个英文数据集, 涵盖了多种类型, 如邮箱、社交网站、游戏论坛等. 其中, 12306、Rootkit 和 ClixSense 包含姓名、邮箱、用户名、生日等在内的 4~6 种个人信息. 鉴于包含个人可识别信息 (PII) 的数据集通常规模较小, 主要用于与 PII 相关的分析和实验. 在进行口令结构和语义特征分析时, 本文将主要使用不包含 PII 的 10 个数据集. 这些数据集在学术界被广泛应用 (见表 1 最右一列).

表 1 本文所使用口令数据集的基本信息

Table 1 Basic information of password datasets used in this paper

口令数据集	服务类型	语言	泄露时间	口令总数	独立口令数量	典型应用文献
126	邮箱	中文	2011-12	6392 568	3778 168	[3, 26, 35]
Dodonew	游戏/电商	中文	2011-12	16 258 891	10 135 260	[28, 49, 50]
CSDN	程序论坛	中文	2011-12	6428 277	4037 605	[22, 51]
Tianya	论坛	中文	2011-12	30 901 241	12 898 437	[52, 53]
12306	购票软件	中文	2014-12	129 303	117 808	[28, 54]
Taobao	电子商务	中文	2016-02	15 072 418	1628 018	[35]
RockYou	社交网络	英文	2009-12	32 603 387	14 326 970	[15, 16, 55]
Yahoo	门户网站	英文	2012-07	442 834	342 510	[22, 56]
LinkedIn	社交平台	英文	2012-01	54 656 615	34 334 121	[15, 23]
000Webhost	网站代管	英文	2015-10	15 299 907	10 583 709	[14, 28]
ClixSense	问卷调查	英文	2016-09	2222 045	1628 018	[26, 34]
Rootkit	黑客论坛	英文	2011-02	69 419	56 900	[3, 28, 56]
Wishbone	社交应用	英文	2020-05	10 092 037	5933 902	[57]

2.2 流行口令

流行口令 (popular password) 是指用户广泛使用的、相对简单且常见的口令. 这些口令通常由短数字、常见单词、常见短语或字符组合构成, 因其普遍性而容易被攻击者利用. 表 2 为不同数据集按频数排序后排名前十的流行口令及其对应的比例. 由于空间限制, 在此仅展示具有代表性的 9 个数据集, 其余数据集具有类似的规律. 可以看出, 除了 CSDN 与 000Webhost 以外, `123456` 在任何数据集中的流行度都占据第一的位置, 不受语言、文化和网站服务类型的影响. 因为口令策略的原因, CSDN 与 000Webhost 的流行口令呈现出与其他数据集略微不同的特点. 例如, CSDN 要求用户生成的口令长度必须大于等于 8; 000Webhost 的口令生成策略为长度至少为 6, 且至少包含一个数字和字母. 为了应对这些策略, 用户会采取一些 "聪明" 的修改, 如 `123456`→`123456789`, `123456`→`a123456`. 这样的修改显然是徒劳的, 攻击者只需对猜测字典进行简单的调整, 即可快速破解出这些 "新的" 流行口令. 此外, 还可以观察到一些流行口令与特定网站或平台密切相关的特点. 例如, 在 Tianya 数据集中, `111222tianya` 排名第 10; 在 Wishbone 中, `wishbone` 排名第 4; 在 CSDN 数据集中, `dearbook` 排名第 4 (dearbook 为 CSDN 旗下书店的域名). 而这些特征都是攻击者所努力挖掘的对象.

表 2 各个数据集中最流行的 10 个口令

Table 2 Top-10 most popular passwords of each dataset

排名	Tianya	126	Dodonew	Taobao	CSDN	000Webhost	Rockyou	Yahoo	Wishbone
1	123456	123456	123456	123456	123456789	abc123	123456	123456	123456
2	111111	12345678	a123456	123456789	12345678	123456a	12345	password	password
3	000000	111111	123456789	111111	11111111	12qw23we	123456789	welcome	123456789
4	123456789	123123	111111	123123	dearbook	123abc	password	ninja	wishbone
5	123123	000000	5201314	000000	00000000	a123456	iloveyou	abc123	12345678910
6	123321	password	123123	5201314	123123123	123qwe	princess	123456789	unicorn
7	5201314	12345678	a321654	aaaaaa	1234567890	secret666	123321	12345678	1234567890
8	12345678	5201314	12345	a123456	88888888	YfDbUfNjH10305070	rockyou	sunshine	12345678
9	666666	a123456	000000	123321	111111111	asd123	12345678	princess	qwertyuiop
10	111222tianya	123321	123456a	7758521	147258369	qwerty123	abc123	qwerty	1234567
百分比	7.43%	3.60%	3.28%	8.74%	10.44%	6.78%	2.05%	1.01%	1.72%

2.3 口令长度分布

表 3 为不同口令数据集的长度分布. 总体看来, 口令长度分布在不同数据集之间存在差异, 但绝大多数用户 (>75%) 的口令长度都集中在 6~11 之间. 除 000Webhost 以外, 所有数据集长度大于 15 的口

令数量都很少 (<1.1%). 此外, 从长度分布中也能反映不同网站的口令策略. 例如, Wishbone 数据集中, 没有长度小于 6 的口令, 这表明, 该网站的口令策略要求用户创建的口令长度必须大于等于 6. 类似地, CSDN 数据集中仅有 2.2% 的口令长度小于 8, 这表明 CSDN 可能在开放注册不久后, 便要求其用户创建口令的长度至少为 8. 除了口令策略, 用户的安全意识也对口令数据集的长度分布有一定的影响. 例如, 000Webhost 的用户多为网站管理员, 其长口令的比例显著高于其他数据集, 即使与具有相似口令策略要求的网站 (CSDN 和 Wishbone) 相比, 000Webhost 长口令的比例也明显更高.

表 **3** 口令长度分布

Table 3 Password length distribution

Dataset	1–5	6	7	8	9	10	11	12	13	14	15	> 15
Tianya	1.79%	33.62%	13.95%	18.08%	9.68%	10.28%	5.59%	2.90%	1.45%	1.33%	0.72%	0.61%
Dodonew	2.43%	12.31%	15.87%	20.87%	22.89%	16.37%	5.21%	1.76%	0.89%	0.56%	0.35%	0.49%
CSDN	**0.63%**	**1.29%**	**0.26%**	36.38%	24.15%	14.48%	9.78%	5.75%	2.61%	2.41%	1.17%	1.09%
126	2.09%	24.02%	17.43%	21.99%	12.82%	9.15%	6.35%	2.60%	1.33%	1.06%	0.61%	0.54%
Taobao	1.14%	12.55%	13.91%	17.64%	18.68%	16.38%	9.52%	5.31%	2.34%	1.40%	0.89%	0.23%
LinkedIn	0.13%	17.46%	12.23%	24.23%	12.35%	9.52%	4.56%	2.83%	1.44%	0.86%	13.77%	0.64%
Rockyou	3.81%	24.45%	18.83%	19.77%	11.73%	9.02%	3.54%	2.08%	1.30%	0.85%	0.54%	0.75%
000Webhost	**0.06%**	5.70%	7.92%	21.87%	15.40%	14.50%	10.49%	**7.66%**	**4.14%**	**3.13%**	**2.10%**	**7.02%**
Wishbone	0.00%	13.50%	8.70%	25.34%	17.69%	13.58%	8.29%	5.50%	3.00%	1.74%	1.69%	0.97%
Yahoo	2.04%	17.89%	14.38%	25.07%	12.41%	11.15%	3.83%	2.59%	1.02%	0.70%	0.45%	0.34%

2.4 口令字符组成和结构

表 4 展示了不同数据集的口令字符组成分布. 观察发现, 中文用户创建口令时, 大约 29%∼64% 的口令是纯数字, 而 10%∼23% 的口令由纯小写字母构成; 英文用户的口令呈现与其相反的趋势, 大约 25%∼42% 的口令是纯小写字母, 仅有 9%∼20% 的口令由纯数字构成. 这意味着数字在中文用户口令中的地位与小写字母在英文用户口令中的地位相当. 这种特性也导致了中英文用户的口令呈现出双相安全性 [52]: 面对在线猜测时 (即允许的猜测次数较小, 如 1∼10 000), 英文用户口令抵御猜测攻击的能力优于中文用户; 面对离线猜测时 (即允许的猜测次数较大, 如 $>10^5$), 中文用户口令抵御猜测攻击的能力优于英文用户.

表 **4** 每个数据集的口令字符组成结构

Table 4 Character composition information about each password dataset

Dataset	[a-z]+	[A-Z]+	[A-Za-z]+	[0-9]+	[A-Za-z0-9]+	[a-z]+[0-9]+	[0-9]+[a-z]+	[a-zA-Z]+[0-9]+	[0-9]+[a-zA-Z]+	[a-z]+1
Tianya	9.91%	0.18%	10.24%	63.77%	98.08%	14.71%	4.12%	15.73%	4.39%	0.12%
Dodonew	10.31%	0.28%	10.90%	30.76%	98.33%	43.51%	7.55%	45.75%	7.93%	1.40%
CSDN	11.64%	0.47%	12.35%	45.01%	96.31%	6.14%	5.89%	28.45%	6.46%	0.24%
Taobao	15.68%	0.17%	16.11%	29.18%	98.93%	45.39%	3.30%	46.26%	3.40%	0.72%
126	22.57%	0.42%	23.78%	42.67%	97.96%	22.72%	4.22%	23.69%	4.39%	1.16%
LinkedIn	28.38%	0.79%	30.68%	24.48%	96.18%	26.18%	2.63%	32.43%	3.04%	3.38%
Rockyou	41.69%	1.50%	44.05%	15.94%	96.25%	27.71%	2.54%	30.18%	2.75%	3.43%
000Webhost	0.05%	0.00%	0.27%	0.03%	93.07%	54.35%	7.28%	60.88%	8.42%	4.66%
Wishbone	25.47%	0.28%	28.50%	9.03%	96.42%	34.30%	1.90%	52.75%	2.79%	2.03%
Yahoo	32.39%	1.70%	35.79%	19.84%	97.99%	27.19%	3.33%	30.79%	3.65%	3.48%

注: 第一行为正则表达式, 如 [a-z]+ 表示仅由小写字母构成的口令 (由于空间限制, 本文在此省去了所有正则表达式开头和结尾的 ^ 和 $ 符号), [0-9]+[a-zA-z]+ 表示数字串后面接小写字母串的口令, [A-Za-z]+ 表示仅由字母构成的口令, [a-z]+1 表示小写字母串后面接数字 1 的口令.

此外, 还有一些有趣的现象值得注意: 无论是中文用户还是英文用户, 都很少 (占比约 0.17%∼1.70%) 采用纯大写字母作为口令; 有不可忽略的用户 (占比约 0.12%∼4.66%) 倾向于在口令末尾添加一个数字 1; 绝大多数用户 (占比 >93%) 的口令都仅由字母和数字组成. 这些特征的分析和挖掘, 将大大缩小攻击者的搜索空间, 进而提高他们的猜测效率.

2.5 基于个人信息构造口令

为了便于记忆, 用户通常在创建口令时会使用自己的个人信息. 然而, 这种行为也带来了潜在的安全风险, 因为攻击者可以利用用户的这一脆弱行为, 大幅提高口令猜测的成功率. 例如, Wang 等人 [28] 的研

究结果表明, 当攻击者已知用户的姓名、生日、用户名、邮箱和手机号时, 仅需进行 100 次猜测, 系统即面临实质性安全风险 (猜测成功率约为 20%).

表 5 为四个不同网站的用户在创建口令时使用个人信息的情况 (其中, PII-Dodonew 数据集是通过邮箱与 12306 数据集匹配获得, 因为 Dodonew 数据集本身不包含个人信息). 观察发现, 两个中文网站的用户更频繁地使用个人信息来构建口令, 而两个英文网站的用户则相对更谨慎. 这一现象极可能与网站的服务类型相关 [52]. 例如, ClixSense 是一家付费问卷调研网站, 与金融相关, 而 Rootkit 是一个黑客论坛, 其用户普遍具有较高的安全意识 (这在文献 [28] 中也得到了确认), 因此这两个网站的用户倾向于更谨慎地使用个人信息构建口令.

表 **5** 用户口令个人信息使用比例

Table 5 Percentages of users building passwords with their own personally identifiable information (PII)

Typical usages of personally identifiable information	PII-ClixSense (2222 045)	PII-Rootkit (69 330)	PII-12306 (129 303)	PII-Dodonew (161 517)
Name (7 subtypes, e.g., `zhangsan`, `zhang`, `szhang`, `zhangs`)	4.85%	4.32%	23.84%	23.82%
Birthday(10 subtypes, e.g., `01171981`, `1981`, `0117`, `19810117`)	7.44%	1.57%	15.61%	16.37%
Email_prefix (3 subtypes, e.g., `moon123`, `moon`, `123`)	5.81%	3.75%	6.86%	8.60%
User name (3 types, e.g., `loveu1314`, `loveu`, `1314`)	5.50%	3.45%	10.51%	10.53%
Phone # (3 subtypes, e.g., `4153022671`, `415`, `2671`)	-	-	0.56%	1.00%
Total personal information usages (all above)	22.30%	12.76%	48.59%	51.43%

总体看来, 虽然不同网站用户之间的口令创建行为存在差异, 但用户在口令创建过程中使用个人信息是一个较为普遍的现象 (12.8%~51.4%). 这也提醒了网站和服务提供商需要采取有效措施来引导用户避免在创建口令时使用个人信息, 以减少潜在的安全风险.

2.6 口令语义分析

为了探究口令所包含的语义信息, 本文以 Wang 等人 [52] 的研究为基础, 考虑 9 种语义模型并构造 4 个语义字典. 它们分别是单字符重复 (如 `111111`)、多字符重复 (如 `123123`)、规律性序列 (如 `654321`)、回文数 (如 `123321`)、键盘模式 (如 `zxcvbn`)、四种日期格式 (完整年月日 YYYYMMDD、部分年月日 YYMMDD、月日 MMDD 和年 YYYY)、英文单词字典、英文姓名字典、拼音单词字典和拼音姓名字典. 表 6 为不同数据集口令所包含的语义信息.

表 **6** 口令中所包含的语义信息

Table 6 Popularity of dictionaries with different semantics in password set

语义模式	126	Dodonew	CSDN	Tianya	Taobao	Rockyou	LinkedIn	000Webhost	Wishbone	Yahoo
单字符重复 (如 `111111`)	4.30%	4.83%	7.42%	7.40%	6.25%	1.47%	2.40%	3.37%	2.06%	2.39%
多字符重复 (如 `123123`)	5.50%	5.16%	7.84%	6.55%	6.07%	3.64%	4.70%	4.03%	4.67%	3.82%
规律序列 (如 `654321`)	9.69%	12.10%	**16.89%**	13.63%	11.47%	5.78%	**7.14%**	**21.01%**	10.67%	6.80%
回文数 (如 `123321`)	3.80%	4.19%	4.05%	3.95%	4.29%	1.95%	3.10%	2.43%	3.00%	2.45%
键盘模式 (如 `zxcvbn`)	5.74%	8.62%	7.16%	6.22%	8.25%	1.52%	2.68%	6.30%	1.84%	2.76%
完整年月日模式 (如 `19990909`)	5.15%	3.88%	8.57%	6.45%	3.14%	0.07%	0.23%	0.22%	0.15%	0.12%
部分年月日模式 (如 `990909`)	10.58%	9.27%	10.09%	16.47%	8.01%	2.37%	4.07%	1.92%	1.78%	2.34%
月日模式 (如 `0909`)	5.34%	5.24%	6.51%	5.47%	5.54%	4.50%	4.21%	2.95%	6.20%	4.94%
年份模式 (如 `1999`)	5.68%	6.66%	6.44%	6.56%	6.00%	2.68%	5.92%	6.53%	7.71%	5.07%
任意日期格式	**26.23%**	**24.76%**	**30.62%**	**34.41%**	**22.47%**	9.06%	13.54%	11.31%	14.99%	11.50%
英文单词字典 (len≥4)	8.56%	3.53%	3.20%	1.87%	3.12%	23.68%	41.99%	13.98%	29.54%	19.48%
英文姓名字典 (len≥4)	8.22%	3.98%	2.53%	1.59%	2.92%	25.83%	30.19%	12.55%	33.58%	18.43%
拼音单词字典 (len≥4)	8.65%	13.94%	14.23%	8.48%	20.57%	3.48%	4.28%	3.62%	3.61%	0.13%
拼音姓名字典 (len≥4)	4.74%	7.73%	7.50%	5.05%	10.80%	1.92%	16.72%	1.84%	1.78%	1.82%

口令与语义字典的匹配方式采用与文献 [23] 相同的最大语义覆盖原则. 例如, 对于口令 `baobaolxy963`, 可以将其拆分为以下几个部分: `baoba`([PinyinWordLower])、`963`([Keyboard]), 和

baobao([PinyinWordLower, SegmentRepeat, PinyinNameAny]). 研究目标是遍历这些可能的语义段 (将语义片段数目记为 n), 确保在语义段数目相同的情况下, 覆盖率最大. 具体而言, 当 $n=1$ 时, 单个语义段 baoba 的覆盖率为 5/12、963 的覆盖率为 3/12、baobao 的覆盖率为 6/12, 最大覆盖率为 6/12; 当 $n=2$ 时, 两个语义段 baoba+963 的覆盖率为 8/12、baobao+963 的覆盖率为 9/12、baoba+baobao 的覆盖率为 6/12, 最大覆盖率为 9/12; 以此类推, 循环退出条件为 n>3 或者已经达到 100% 覆盖率.

观察发现, 不论是中文用户还是英文用户, 其口令中都包含了丰富的语义信息. 这些信息可以分为多种模式, 包括字符重复、规律序列、日期、键盘模式和语义字典 (中文用户的口令对应拼音字典, 英文用户的口令对应英文单词字典). 其中, 中文用户更倾向于在口令中使用日期信息, 尤其是年月日模式 (如 990525), 这可能与个人生日或重要纪念日有关, 也使得这部分用户的口令更容易遭受定向攻击. 值得注意的是, 中英文用户在创建口令时都倾向于使用规律的序列 (如 654321, abcdefg), 这种现象在有口令策略要求的网站中尤为突出. 例如, 在中文数据集中, CSDN 口令的规律序列占比最高 (16.89%); 在英文数据集中, 000Webhost、Wishbone 和 LinkedIn 中规律序列的占比排名前三.

2.7 口令重用

随着各种数字化网络服务的迅速发展, 用户需要管理的账户数量不断增加. 研究表明, 通常情况下, 互联网用户需要管理约 100 个口令 [58]. 然而, 由于人类的记忆能力有限且难以稳定地记住如此多的口令, 用户不可避免地倾向于重复使用口令或对现有口令进行简单的修改 (即间接口令重用). 频繁发生的大规模口令数据泄露事件, 为攻击者提供了丰富的数据资源. 一旦攻击者成功获取了用户在一个网站的口令, 就很容易攻破其重用相同或相似口令的其它账户.

Wang 等人 [3] 在 2016 年对口令重用相关的研究进行了总结: 约 21%∼51% 的用户会直接重用已有口令, 有 26%∼33% 的用户会间接重用口令. 在此, 间接重用的衡量标准为新旧口令的相似度在 [0.8,1] 之间, 其使用的相似度指标为基于文氏距离、曼哈顿距离的文本相似度算法. 以大规模口令泄露合集 COMB [13] 为例 (包含超过 32 亿独立的邮箱和明文口令对), 本文对用户多个口令的泄露现状进行了统计. 具体而言, 使用邮箱前缀进行匹配 (例如, zhangsan123@google.com 与 zhangsan123@yahoo.com 的前缀一致, 则认为这两个邮箱对应同一用户), 共计得到约 14 亿数据条目, 其中 16% 的邮箱前缀对应 2 个口令, 6% 的邮箱前缀对应 3 个口令, 10% 的邮箱前缀对应至少 4 个口令.

更进一步, 本文从 (通过邮箱前缀匹配得到的) 口令数目大于等于 3 的数据条目中 (共计约 2.2 亿), 随机选取 3 个口令 (分别记为 pw_A, pw_B 和 pw_C), 探究用户多口令重用的现状. 特别地, 以基于编辑距离的相似度算法为例 (相似度计算方式为 $s=1-\mathrm{EditDistance}(\mathrm{pw}_A,\mathrm{pw}_B)/\max(|\mathrm{pw}_A|,|\mathrm{pw}_B|)$, 相似度阈值设为 0.5), 本文统计了用户 3 个口令之间的相似关系, 详细结果如表 7 所示.

表 **7** 用户口令重用统计

Table 7 Statistics about password reuse

数量		描述	比例 (%)
使用邮箱前缀匹配	1430 044 011	相同邮箱前缀对应 2 个口令	16
		相同邮箱前缀对应 3 个口令	6
		相同邮箱前缀对应至少 4 个口令	10
用户口令相似性 (口令数量≥3)	220 316 017	第一个口令与第三个口令相同 ($\mathrm{pw}_A=\mathrm{pw}_C$)	33.80
		第一个口令与第三个口令相似 ($\mathrm{pw}_A\approx\mathrm{pw}_C$)	49.08
		第二个口令与第三个口令相同 ($\mathrm{pw}_B=\mathrm{pw}_C\&\mathrm{pw}_A\neq\mathrm{pw}_B$)	10.36
		第二个口令与第三个口令相似 ($\mathrm{pw}_B\approx\mathrm{pw}_C\&\mathrm{pw}_A\neq\mathrm{pw}_B$)	9.70
		三个口令相似但不相同 ($\mathrm{pw}_A\approx\mathrm{pw}_B\approx\mathrm{pw}_C$)	4.10

观察发现, 约有 33.80% 的用户的第一个口令与第三个口令完全相同 ($\mathrm{pw}_A=\mathrm{pw}_C$). 此外, 约 10.36% 的用户的第二个口令与第一个口令不同 ($\mathrm{pw}_A\neq\mathrm{pw}_B$), 但与第三个口令完全相同 ($\mathrm{pw}_B=\mathrm{pw}_C$). 这表明, 攻击者利用用户泄露的旧口令 pw_A 直接攻击目标口令 pw_C 的成功率为 33.8%; 若攻击者再直接利用同一用户额外泄露的第二个旧口令 pw_B, 即可多增加 10% 的猜测成功率. 其次, 还发现大约 4.1% 的用户的三个口令之间相似但不相同. 这表明这些用户在创建多个口令时可能存在一定的规律模式. 因此, 需要设

计新的模型来更深入地理解用户多个旧口令之间的内在联系.

3 口令猜测攻击

口令猜测攻击根据技术路线的不同, 大致可分为三种类型. 第一类为基于规则的猜测攻击, 此类方法通过设计各种口令变换规则, 然后应用于已有口令或单词, 更多依赖于攻击者的经验. 第二类为基于统计学方法的猜测攻击, 例如, 基于概率上下文无关文法 (PCFG [21])、基于马尔科夫链的方法 (Markov [20,22]). 这些方法使用数学和统计模型来分析和生成口令, 可以考虑上下文语法规则, 提高了口令猜测的成功率. 第三类为基于机器学习的猜测攻击, 近年来这一领域得到了广泛的关注. 学者们尝试采用各种复杂的神经网络结构 (如 RNN [14]、GAN [15]、WAE [16]) 来解决传统统计学方法存在的数据稀疏和过拟合问题. 此外, 还有研究 (如文献 [24]) 将机器学习技术与开源工具 (如 Hashcat [19]) 相结合, 大大提升了原始工具的猜测效率. 下面将对这三类方法中具有代表性的算法/工具/模型/框架进行介绍. 图 1 和表 8 对 2005 年以来的具有代表性的口令猜测算法进行了分类和总结.

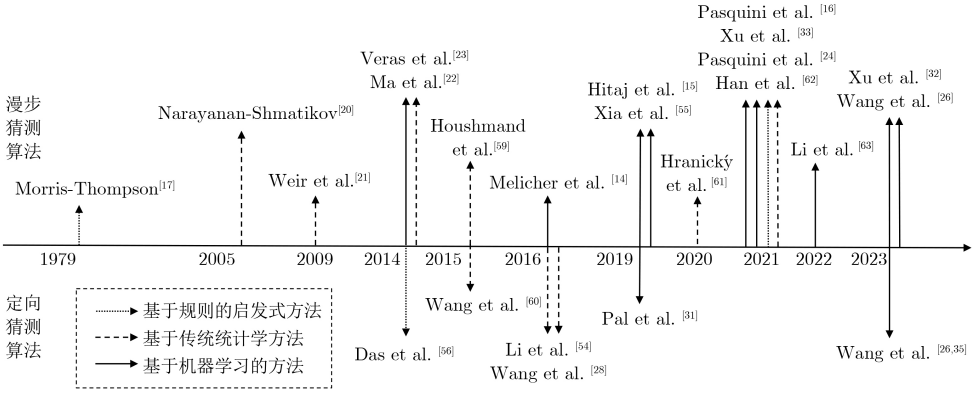


图 1 口令猜测算法的发展脉络和分类

Figure 1 Evolutionary trends and classification of password guessing algorithms

3.1 基于规则的攻击方法

3.1.1 启发式算法

不同于传统的暴力破解方法, 启发式算法很大程度上依赖于攻击者的经验. 其核心思想是观察用户在创建口令时的脆弱行为, 然后运用各种启发式技巧以提高口令猜测的效率和成功率. 其中一种常见的技巧是基于规则的变换, 攻击者会利用口令中常见的规律和模式, 对已有口令或单词进行多种变换, 例如首字母大写、尾部字符 (串) 的插入/删除、字符替换 (如将字符 `a` 替换为字符 `@`)、字符串反转等. 早期, 仅有一些欧美机构进行了零星的研究. 例如, 1979 年, Morris 和 Thompson [17] 设计了各种启发式变换规则, 用于生成字典中单词的变种, 以进行口令猜测; 1990 年, Klein [64] "精心设计" 了口令猜测的顺序, 用于攻击 15 000 个经过 DES 加密的真实口令; 1999 年, Thomas [65] 基于用户群体的领域特征构建了口令猜测字典, 用于攻击 2500 个经过 DES 加密的真实口令.

尽管此类方法缺乏严格的理论基础和系统性, 但在特定场景下, 如果设计合理, 其猜测成功率也不容小觑. 例如, 2014 年, Das 等人 [56] 提出了一种基于用户旧口令和规则变化的启发式口令猜测算法. 该算法按照预定义的规则顺序对用户已有的原始口令应用八类变换操作 (如插入、删除、大写等). 在每个变换阶段之后, 检查生成的口令是否已经猜测到了目标口令. 在小猜测数下 (如 100 次), 该算法的猜测成功率达到 30%, 远优于漫步猜测算法, 但它也存在一些固有的缺陷: 它假设所有用户都以固定的优先级选择口令变换规则, 这不符合用户重用口令的真实行为; 生成口令猜测时, 该算法仅考虑了一种规则变换情况, 没有考虑两种或更多规则变换的组合情况, 这无法捕捉用户重用口令时的复杂修改行为.

表 **8** 主要口令猜测算法总结

Table 8 Summary of mainstream password guessing methods

算法名称	技术原理	猜测场景		核心贡献
		漫步	定向*	
Markov, 2005 [20]	Markov	✓		首次将 Markov 模型引入口令猜测
Markov, 2014 [22]	Markov	✓		为 Markov 模型引入了一系列平滑与正规化技术
OMEN, 2015 [66]	Markov	✓		优化了 Markov 模型的口令生成过程, 提高了口令生成速度
Tar-Markov, 2015 [3,60]§	Markov		✓	首次将个人信息标签引入 Markov 模型
Chunk_level Markov, 2021 [33]	Markov	✓		利用 BPE 算法将口令字符序列表示为 chunk (如`4ever`) 序列
PCFG, 2009 [21]	PCFG	✓		首次提出基于 PCFG 文法的口令猜测模型
Veras et al., 2014 [23]	PCFG	✓		对原始 PCFG 的字母 L 段进行语义挖掘
Houshmand et al., 2015 [59]	PCFG	✓		加入键盘与多词模式
TransPCFG, 2021 [62]	PCFG	✓		利用短口令获取的 PCFG 文法信息, 对猜测长口令进行改进
Personal-PCFG, 2016 [54]	PCFG		✓	增加了基于长度的个人信息标签 (如姓氏 `wang` 表示为 N_4)
TarGuess-I, 2016 [28]	PCFG		✓	增加了基于类型的个人信息标签 (如姓氏 `wang` 表示为 N_1)
TarGuess-II, 2016 [28]	PCFG		✓	首个针对口令重用的概率模型, 主要分为结构级和字段级变换
TarGuess-III, 2016 [28]	PCFG		✓	可同时利用用户个人信息和旧口令
TarGuess-IV, 2016 [28]	PCFG		✓	首次引入隐式个人信息 (如性别)
Hranický et al., 2020 [61]	PCFG	✓		将部分生成的句法形式分布到多节点, 减少网络传输数据量
Chunk_level PCFG, 2021 [33]	PCFG	✓		改进了 PCFG 的结构段划分模式 (如 p@ss 表示为 DM_4)
Das et al., 2014 [56]	Mangling Rule		✓	设计了 8 类启发式规则变换进行口令重用攻击
MDBSCAN, 2022 [63]	MR+ML†	✓		对口令进行无监督聚类, 再生成排序的规则集
AdaMs, 2021 [24]	MR+DL†	✓		利用神经网络建模口令与规则的匹配度
FLA, 2016 [14]	LSTM	✓		首次将循环神经网络引入口令猜测
Chunk_level FLA, 2021 [33]	LSTM	✓		利用 BPE 算法将口令字符序列表示为 chunk (如`4ever`) 序列
PassGAN, 2019 [15]	GAN	✓		首次将生成式对抗网络引入口令猜测
Pass2Path, 2019 [31]	Seq2Seq		✓	利用旧口令序列生成对应新口令的编辑操作序列
GENPass, 2019 [55]	PCFG+LSTM	✓		利用 LSTM 生成口令结构段
CPG/DPG, 2021 [16]	WAE/GAN	✓		引入自编码模型, 改进 PassGAN, 提出动态调整机制
Pass2Edit, 2023 [35]	GRU		✓	利用多步决策机制, 将口令重用行为建模为多分类问题
RFGuess, 2023 [26]‡	Random Forest	✓	✓	对口令字符重新编码, 引入经典机器学习技术
PassBERT, 2023 [32]††	Transformer	✓	✓	引入微调技术, 利用双向 Transformer 进行口令猜测

* 定向猜测场景包含两类, 分别为基于个人信息 (PII) 的定向猜测和基于用户旧口令 (口令重用) 的定向猜测.

§ 该方法最早由文献 [60] 提出, 仅使用了姓名信息; 在文献 [3] 中, 作者进一步拓展使用了生日、邮箱等 6 类个人信息.

† MR+ML 表示 Mangling Rule 与 Machine learning 结合; MR+DL 表示 Mangling Rule 与 Deep learning 结合.

‡ RFGuess 为基于随机森林的口令猜测框架, 包含三个口令猜测算法, 分别是漫步口令猜测算法 RFGuess、基于个人信息的定向口令猜测算法 RFGuess-PII 和基于口令重用的定向口令猜测算法 RFGuess-Reuse; 详细描述见第 3.3.7 节.

†† PassBERT 为基于双向 Transformer 的口令猜测框架, 包含三个口令猜测算法, 分别是条件口令猜测算法 PassBERT(-CPG)、基于口令重用的定向猜测算法 PassBERT(-TPG) 和基于规则匹配度的算法 PassBERT(-ARPG); 详见第 3.3.8 节.

3.1.2 口令破解工具

Hashcat [19] 和 John the Ripper(JtR) [18] 是两款最为流行的口令破解工具, 它们在学术研究和实际应用中都发挥着重要作用. 这两个工具包的运行效率经过高度优化, 能充分发挥多核处理器和 GPU 提供的并行计算性能. 它们提供多种攻击模式, 如掩码攻击、规则变换和字典组合等, 将传统的纯暴力破解攻击转化为更为智能和有针对性的搜索策略. 虽然 Hashcat 和 JtR 各自有一些独特的指令/模式, 但这两个工具共享几乎相同的规则变换方法. 值得注意的是, 它们在应用规则变换时的策略略有不同. 具体而言, Hashcat 遵循以单词为主的顺序 (word-major order), 即在考虑下一个字典中的单词之前, 会将规则集中的所有规则应用于当前单词. 相反, JtR 遵循以规则为主的顺序 (rule-major order), 即在应用下一个规则之前, 将当前规则应用于字典中的所有单词. 这种差异对于研究基于规则的口令破解方法具有重要意义.

尽管学术界提出了一系列口令猜测算法, 并声称其猜测成功率优于口令破解工具 Hashcat/JtR, 但这些比较实验往往采用了破解工具的默认配置, 无法反映经验丰富的攻击者的真实能力. 2015 年, Ur 等人 [67] 调研了研究人员常用的口令猜测算法与专业攻击者实际采用策略之间存在的差异, 强调了仅仅依赖单一破解算法作为口令安全性度量存在偏差, 并首次提出了结合各类猜测算法/工具的理想指标 Min_auto. 该指标可理解为采用同一口令在不同猜测方法下猜测数的最小值作为其安全性的度量.

事实上, 经验丰富的攻击者通常会精选合适的字典和规则, 并根据中间结果和破解成功率灵活地调整攻击策略. 例如, 在文献 [68] 中, 详细描述了三名攻击者如何利用专业经验快速破解约 90% 的口令. 在此, 以第一位攻击者 (记为攻击者 $\mathcal{A}$) 的破解策略为例, 进行简要介绍.

攻击者 $\mathcal{A}$ 的破解策略可以分为三个主要阶段, 每个阶段采用不同的攻击方法. 以下是对这三个阶段的简要概述: 在第一阶段, 他采用了包括暴力破解和基于字典的攻击在内的多种攻击策略. 具体而言, 他首先尝试了 1 到 6 位长度的所有字符组合, 包括大小写字母、数字和特殊字符. 然后, 他分别对 7 或 8 位长度的纯小写字母和纯大写字母进行破解尝试. 接着, 他尝试了 1 到 12 位长度的纯数字组合. 此外, 他还应用了一个多年来精心筛选的字典, 并结合了 best64 规则集进行攻击. 在已破解的口令的基础上, 他还使用了 d3ad0ne 规则集进行进一步尝试. 在此阶段, 攻击者 $\mathcal{A}$ 成功破解了 62% 的口令, 仅耗时 16 分钟. 在第二阶段, 攻击者 $\mathcal{A}$ 采用了混合攻击策略, 结合了字典和掩码. 他依次尝试在字典中每个口令的末尾添加了 2 个字符 (数字或特殊符号)、3 个字符 (数字或特殊符号)、4 个数字, 以及 3 个小写字母和数字的组合. 在此阶段, 攻击者 $\mathcal{A}$ 再次成功破解了 16% 的口令, 过程耗时 5 小时 12 分钟. 最后, 在第三阶段, 攻击者 $\mathcal{A}$ 采用了 hashcat 掩码模式内置的马尔科夫优化, 将先前成功破解的口令用作训练集, 并结合自定义字典和规则, 进一步成功破解了 10% 的口令, 耗时 14 小时 59 分钟. 需要注意的是, 这一破解流程具有明显的阶段性, 每个阶段的难度逐渐增加, 其根本原因在于口令的分布服从 Zipf 分布 [49,57]: 流行的口令和罕见的口令, 都占据用户群体中相当大的比例.

近年来, 陆续有研究者对如何更合理/高效地利用口令破解工具展开了研究. 例如, 在 2019 年, Liu 等人 [69] 介绍了一种分析和高效处理破解工具 (如 Hashcat 和 JtR) 的技术. 作者提出了两种新的操作, 即规则反转 (rule inversion) 和猜测计数 (guess counting), 通过这些操作, 他们能够快速分析口令在此类工具下的安全强度而无需列举所有可能的猜测; 在 2021 年, Zhang 等人 [70] 设计了一个实用的口令数字语义提取工具, 基于经验性分析, 他们提出了两种新的操作 (operations, 即构建规则集的基本单位), 并生成了 1974 个数字语义规则. 通过对 JtR 和 Hashcat 规则引擎和运行逻辑的优化, 作者实现了 JtR 和 Hashcat 对所提出新操作的支持; 在 2021 年, Pasquini 等人 [24] 利用深度学习技术对口令与规则的适配度进行了优化, 提出了基于规则的自适应攻击算法 ADaMs (详细描述见第 3.3.5 节); 2022 年, Di Campi 等人 [71] 对规则依据频数进行排序, 发现使用大规模字典结合少量高效的规则更为有效; 同年, Li 等人 [63] 采用无监督聚类方法生成更为有效的口令规则集, 使得字典攻击的猜测成功率在大猜测数下高于现有的口令猜测算法. 具体而言, 作者首先使用 Damerau-Levenshtein 距离和 Jaro-Winkler 距离作为度量指标, 运用改进的无监督聚类算法 DBSCAN 对口令进行聚类. 在每个聚类中, 他们选择了与所有其他元素的距离之和最小的字符串作为基准字符串 (base string). 接着, 利用 Damerau-Levenshtein 距离算法, 从基准字符串到目标字符串生成了一系列插入、删除和替换操作, 形成单个类别的规则集. 最后, 他们将从所有聚类中生成的规则集合并, 并按照规则出现的频数进行排序, 再通过 Hashcat [19] 工具, 将基础字典与规则集结合生成最终的猜测集.

3.2 基于统计学的攻击算法

3.2.1 PCFG

2009 年, Weir 等人 [21] 提出了一个基于概率上下文无关文法的漫步口令猜测算法 PCFG. 该算法将口令字符分为三种类型: 字母段 L、数字段 D 和特殊字符段 S, 并假设不同类型的字段之间相互独立. 它将口令按照不同字符类型和对应子串长度转化为中间结构 (模板). 例如, 口令 `password123!` 可以表示为 $L_4D_3S_1$, 其中 L_4 代表长度为 4 的字母段 `wang`, D_3 代表长度为 3 的数字段 `123`, S_1 代表长度为 1 的特殊字符段`!`. 其训练和生成过程简要描述如下 (如图 2 所示): 首先, 从训练集中统计出各类口令结构 (如 $L_4D_3S_1$) 及其实例化 (如 L_4→`wang`) 所对应的概率; 然后, 通过从训练数据集中学习到的相同长度的实例化 (instantiations) 替换这些 “L、D、S” 标签, 生成候选口令及其对应的概率; 最后, 对生成的口令依照概率大小排序形成猜测集. 例如, 口令 `wang123!` 在 PCFG 模型下的概率可以计算为 $\Pr(\texttt{wang123!}) = \Pr(L_4D_3S_1) \times \Pr(L_4 \to \texttt{wang}) \times \Pr(D_3 \to 123) \times \Pr(S_1 \to \texttt{!})$.

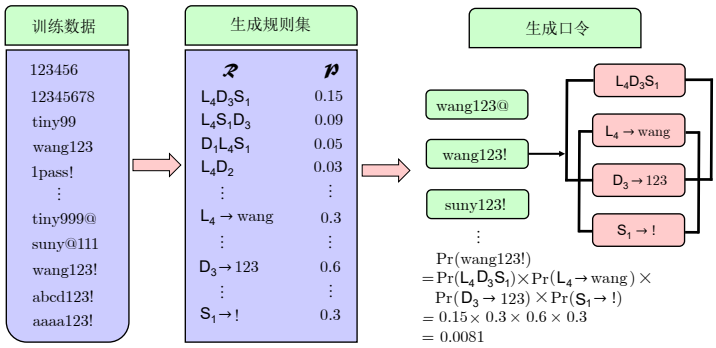


图 2 PCFG 训练与生成过程示意图
Figure 2 Training and generation process of PCFG

PCFG 受到后续研究者的持续关注, 学者们不断扩展和优化其在口令猜测领域的应用, 衍生出一系列基于 PCFG 的改进/新型猜测算法. 例如, 2014 年, Veras 等人 [23] 对口令中包含的语义信息进了深入挖掘, 提出了融合语义的 PCFG 算法. 该方法主要分为三步:

首先, 借助语料库对口令进行分词处理. 例如, 口令 `loveblue0507` 分词后的结果是 `love`、`blue` 和 `0507`. 在其中, 字母构成的部分被称为 “word 段”, 而数字和特殊字符构成的部分被合称为 “gap 段”.

其次, 对这些 “word 段” 进行词性标注, 即为每个字段赋予一个相应的词性类别. 举例来说, `blue` 被标注为名词单数 (NN). 接下来, 进行词性分类与泛化操作. 对于 “word 段”, 如果该段所属词汇在源语料库中, 那么它会被划分至相应的语料库类别 (如 “月份”、“女性名字” 等). 否则会利用 Wordnet (一种同义词集合工具) 将其类别归纳为一个通用的专业表达形式. 例如, 将 `love` 分类为 love.n.01, 表示喜爱的第一个名词含义; 词性分类完成后, 需要构建一个树模型进行抽象泛化, 例如, `soldier` 的语义标签为 worker.n.01 (“worker” 的第一个名词义项), 其泛化后的结果为 soldier.n.01.

最后进行猜测生成, 该过程和原始 PCFG [21] 类似. 例如, 生成口令 `youlovethem2` 的概率计算公式为: $\Pr(\texttt{youlovethem2}) = \Pr(N_1 \to [\text{PP}][\text{love.v.01.VV0}][\text{PP}][\text{number}]) \times \Pr([\text{PP}] \to \texttt{you}) \times \Pr([\text{love.v.01.VV0}] \to \texttt{love}) \times \Pr([\text{PP}] \to \texttt{them}) \times \Pr([\text{number}] \to 2)$.

2014 年, Ma 等人 [22] 发现, 利用原始训练集的实例化 (instantiations) 来填充 PCFG 的字母段 L, 其效果通常要优于从外部引入字典; 2015 年, Houshmand 等人 [59] 通过向上下文文法中引入键盘模式 (如 `qwerty`) 和多词模式 (如 `boatboat`) 改进了原始 PCFG 的破解成功率; 2019 年, Wang 等人 [52] 对规律性的长结构 (如 $D_1L_1D_1L_1D_1L_1$) 进行了优化, 并引入拼音和姓名字典至原始 PCFG 的字母段 L, 改善了 PCFG 在中文数据集上的破解成功率; 同年, Zhang 等人 [50] 利用改进的 Markov 模型来生成 PCFG 中 L、D、S 段中的填充内容, 提升了原始 PCFG 的泛化能力; 2020 年, Hranický 等人 [61] 提出了一种使用概率文法进行分布式破解口令的技术. 其核心思想是将部分生成的句法形式分布到多个节点上, 从而减

少了需要在网络中传输的数据量, 提高了破解口令的效率; 2021 年, Han 等人 [62] 提出了一种面向长口令的 TransPCFG 框架. 其核心原理在于从基于短口令训练的 PCFG 模型 (简称 PCFGshort) 中提取有效的语法信息, 并将这些信息传递到基于长口令训练的 PCFG 模型 (简称 PCFGlong) 中. 同时, 还根据 PCFGshort 中的结构信息来动态调整 PCFGlong 中的结构信息, 从而提高了 PCFG 针对长口令的破解成功率; 此外, 还有研究人员将个人信息标签 (如姓名、生日、用户名、邮箱、手机号等) 引入 PCFG, 提出了基于 PCFG 的定向口令猜测算法, 如文献 [28, 54].

3.2.2 Markov

在 2005 年, Narayanan 和 Shmatikov [20] 首次将马尔可夫模型 (Markov) 引入到口令猜测. Markov 模型 (又称 n-gram 模型) 有阶的概念, d 阶 Markov($d = n + 1$) 的核心假设是, 当前口令字符仅与和它对应的前 d 个字符有关, 与其它字符无关. 例如, 3 阶 Markov 模型对口令 `abc123` 的概率计算如下: $\Pr(\texttt{abc123}) = \Pr(\texttt{a}) \times \Pr(\texttt{b}|\texttt{a}) \times \Pr(\texttt{c}|\texttt{ab}) \times \Pr(\texttt{1}|\texttt{abc}) \times \Pr(\texttt{2}|\texttt{bc1}) \times \Pr(\texttt{3}|\texttt{c12})$, 其中 $\Pr(\texttt{3}|\texttt{c12})$ 的计算方式为字符串 `c12` 后是 `3` 的频数除以 `c12` 后有字符的频数, 其余条件概率的计算方式与之相同.

当 Markov 模型的阶数过大时, 会出现数据稀疏的问题. 例如, 当阶数为 5 时, 若训练集中没有出现 `abcde1` 这样的字符串, 那么转移概率 $\Pr(\texttt{1}|\texttt{abcde})$ 的计算结果为 0. 此外, 原始 Markov 模型所生成的所有口令的概率加和不等于 1.0. 例如, 若训练集为 {`ab`,`abc`}, 那么 1 阶 Markov 模型计算口令 `ab` 的概率为 $\Pr(\texttt{ab}) = \Pr(\texttt{a}) \times \Pr(\texttt{b}|\texttt{a}) = 1$, 如果该模型再生成其他口令, 概率总和显然将大于 1.0.

为了解决这些问题, 2014 年, Ma 等人 [22] 提出了一系列平滑 (如 Laplace 平滑) 和正规化方法 (如 End-symbol 方法), 实验结果表明使用 End-symbol 方法的 Backoff 模型表现最优, 在此对其进行简要介绍. End-symbol 是指在每个口令的末尾增加一个终止符号 (记为 $\perp$), 在训练时, 将其视为普通字符纳入统计; 在生成时, $\perp$ 可以出现在除了开头以外的任何位置, 标志着单个口令生成完毕. 仍以 {`ab`, `abc`} 为例, 在对每个口令增加终止符后, 1 阶 Markov 模型计算口令 `ab` 的概率变为 $\Pr(\texttt{ab}) = \Pr(\texttt{a}) \times \Pr(\texttt{b}|\texttt{a}) \times \Pr(\perp|\texttt{b}) = \frac{1}{2}$. 图 3 为使用正规化方法的 Markov 模型的训练和生成过程示意图. Backoff 是一种变阶 Markov 模型, 当某个字符所对应的 d 阶前缀的频数小于某个阈值时, 模型启用回退 (backoff) 机制, 使用该字符对应的更短的前缀 (如 $d-1$ 阶前缀), 直至该字符所对应前缀的频数满足阈值. 例如, 当阶为 4, 计算转移概率 $\Pr(\texttt{1}|\texttt{wang})$ 时, 如果字符串 `wang1` 的频数不满足阈值, 而 `ang1` 的频数满足阈值, 则计算转移概率 $\Pr(\texttt{1}|\texttt{ang}) \times \omega(\texttt{wang})$ 来代替原始的 $\Pr(\texttt{1}|\texttt{wang})$, 其中 $\omega(\texttt{wang})$ 是一个与所有以 `wang` 开头的长度为 5 的字符串有关的值, 作用是将概率归一化, 使得整个口令空间的概率总和仍为 1.0.

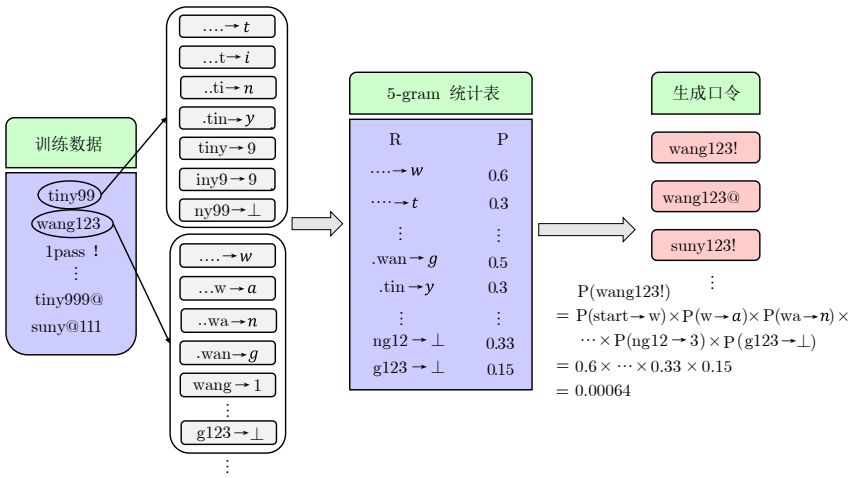


图 3 Markov 训练与生成过程示意图

Figure 3 Training and password generation process of Markov model

2015 年, Dürmuth 等人 [66] 对原始的 Markov 模型的口令生成过程进行了优化, 提出了 OMEN 模型, 其核心原理在于将 Markov 模型中字符的概率值映射为等级, 使得模型可以 (在整体层面) 按照概率降

序生成候选口令, 从而大大提升了生成口令猜测的速度.

3.2.3 基于 PCFG 的定向口令猜测框架 TarGuess

2016 年, 基于 PCFG [21]、Markov [22] 和贝叶斯理论等统计计算理论, Wang 等人 [28] 提出了一个定向口令猜测理论框架 TarGuess, 涵盖 4 种基于不同个人信息组合的定向猜测攻击算法和 3 个理论攻击模型. 具体而言, 他们根据个人信息在构造口令时的作用和隐私程度, 将个人信息分为三类. 第一类为能够直接用于构造口令的显式 PII (称为 Type-1 PII), 如姓名和生日; 第二类为隐式 PII (称为 Type-2 PII), 如性别和受教育程度等, 它们虽然不能直接作为口令的一部分, 但它们会影响用户的口令创建行为; 第三类为用户的认证凭证信息, 如口令. 根据攻击者所获取的个人信息的不同, TarGuess 框架主要刻画了四种最具代表性的定向猜测场景: (1) 已知用户的 Type-1 PII; (2) 已知用户已泄露的旧口令; (3) 同时已知用户的个人信息和已泄露的旧口令; (4) 在场景 #3 的基础上, 额外使用了目标用户的 Type-2 PII (如性别). TarGuess 猜测框架图如图 4 所示.

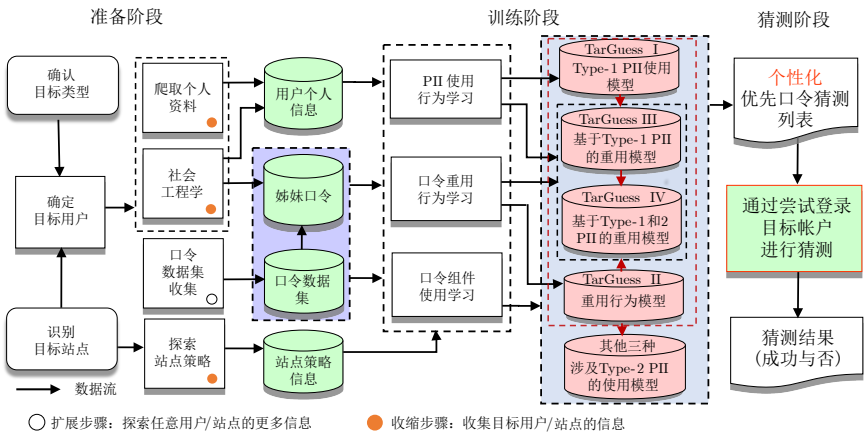


图 4 TarGuess 猜测框架图 [28]

Figure 4 An architectural overview of TarGuess [28]

以 TarGuess-I [28] 为例, Wang 等人在 PCFG [21] 原有的字母 L、数字 D 和特殊符号 S 标签的基础上, 引入了基于类型的个人信息标签 (如姓名、生日、邮箱、用户名等). 例如, 使用 N 代表姓名信息, N1 表示姓名全称, N2 表示姓名全称的缩写, N3 表示姓氏, $\cdots$; 使用 E 表示邮箱信息, E1 表示邮箱前缀的全称, E2 表示邮箱前缀的第一个字母段, $\cdots$. 不难发现, 新标签与原有的 L、D、S 标签的不同之处在于, 其下标数字不再代表长度, 而是代表不同的类型. 通过这种表示方式, 可以有效避免使用长度划分个人信息 [54] 所带来的对字段的子类型不敏感的问题.

TarGuess-II 综合考虑了结构级 (如 $\mathsf{L}_3\mathsf{D}_3\rightarrow\mathsf{L}_3\mathsf{D}_3\mathsf{S}_2$) 和字段级 ($\mathsf{L}_3$:abc$\rightarrow\mathsf{L}_4$:abcd) 两种口令修改变换方式, 并借鉴图同构的概念, 将新旧口令的结构级修改行为映射为传统 PCFG 的结构变换. 在此基础上, 利用 Markov 模型来模拟字段内部的修改过程, 最终构建出一个完整的口令变换上下文无关文法模型. TarGuess-III 在 TarGuess-II 的基础上, 引入了基于类型的 PII 标签 (与 TarGuess-I 一致), 构建了一个可识别 PII 的口令重用文法. TarGuess-IV 则通过引入贝叶斯理论, 将性别这一无法直接用于构造口令的隐式个人信息融入猜测模型, 其本质是对生成的猜测进行概率调整.

2020 年, Xie 等人 [72] 在 TarGuess-I [28] 原有个人信息标签类型的基础上, 增加了键盘模式、流行口令标签和特殊字符串标签, 提出了 TarGuess-I$^+$, 进一步提升了原始 TarGuess-I 的猜测成功率.

3.3 基于机器学习技术的攻击算法

3.3.1 基于循环神经网络的口令猜测模型

2016 年, Melicher 等人 [14] 首次将深度学习技术用于口令猜测, 提出了基于循环神经网络 (RNN) 的口令猜测模型 FLA. 该模型由 3 层长短期记忆网络 (LSTM) 和两层全连接层构成. 其核心思想是通过

口令中先前的字符序列来预测下一个字符出现的概率, 本质上是一个字符级的语言模型. 与 Markov 模型 [22] 的不同之处在于, FLA 模型在输入字符前缀后, 能自动为字符表中的每个字符都分配一个非零的概率, 而不是通过启发式的平滑技术 (如 Laplace 平滑). 这使得 FLA 相比 Markov 模型具有更好的泛化性.

2019 年, Xia 等人 [55] 将 LSTM 与 PCFG 结合, 提出了口令猜测模型 PL(PCFG+LSTM). 该模型首先将训练集中的口令转换成结构段的形式 (如 `password123!`→$\mathsf{L}_8\mathsf{D}_3\mathsf{S}_1$), 再将结构段序列用于 LSTM 网络的训练; 模型训练完成后, 可生成一系列结构段序列, 再使用与原始 PCFG 一致的方式对这些结构段进行相应填充. 为了在不同测试集中能达到普遍较优的猜测效果, 作者进一步训练了多个 PL 模型, 将这些模型的输出 (即结构段及其对应概率) 相结合, 并进行权重选择, 再通过原始口令数据集训练的 CNN 分类器判断新生成口令的来源, 最后通过设计的阈值判断生成口令是否是通用的 (general), 作为接受和拒绝生成口令的依据. 此过程构成了多源口令猜测方法 GENPass. 2021 年, Wang 等人 [73] 在 PL 模型的基础上, 利用循环神经网络 (RNN) 来生成 PCFG 的结构段以及 L 段的填充内容, 提出了 PR+ 模型, 降低了模型对大训练样本的依赖.

3.3.2 基于生成式模型的口令猜测算法

2019 年, Hitaj 等人 [15] 首次将生成对抗网络 (GAN [74]) 引入到口令猜测领域, 提出了 PassGAN 模型, 它由生成器 G 和判别器 D 构成. 不同于 FLA 模型 [14] 的自回归生成方式, PassGAN 学习整个口令的字符分布. 在训练过程中, 生成器 G 接收随机噪声向量作为输入, 旨在生成与真实口令样本相似的假口令样本. 判别器 D 对来自训练数据集的真实口令样本和生成器生成的假口令样本进行区分, 从而提供反馈. 生成器 G 根据判别器 D 的反馈逐步调整模型参数, 生成更逼真的假口令样本, 使其分布逐渐接近训练集中真实口令样本的分布. 在训练完成后, 只需不断向生成器 G 提供噪声向量, 就能生成口令, 直到达到所需的猜测集大小. 图 5 为 GAN 用于口令猜测的示意图. 不同于传统方法 (如 PCFG [21] 和 Markov [22]) 通过概率统计建立精确的口令概率分布模型, 基于 GAN 的口令猜测模型无法为生成的口令样本赋予概率并排序. 实验表明 [15], 要达到与 PCFG 和 Markov 相同的猜测成功率, PassGAN 至少需要生成高一个数量级的猜测. 尽管该方法的猜测成功率不如 PCFG 和 Markov 模型, 它显示了 GAN 在生成口令方面的巨大潜力, 也为后续研究提供了新的思路和方向.

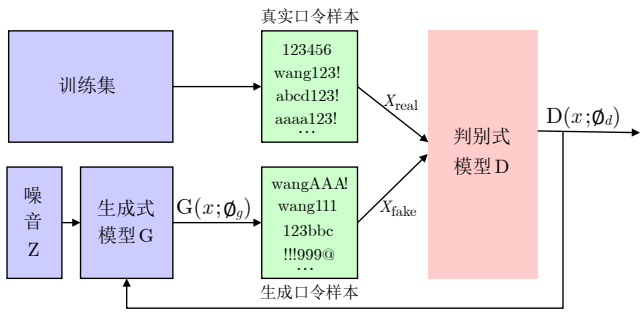


图 5 基于 GAN 的口令猜测模型

Figure 5 GAN-based password guessing model

2021 年, Pasquini 等人 [16] 指出, 口令的离散表征使得生成器在训练过程中难以正确地反映数据分布, 从而导致训练过程的不稳定性; 此外, 生成器不能很好地学习训练数据的离散特性, 使得判别器可以轻易区分真实数据和生成数据, 限制了生成器的改进空间. 为此, Pasquini 等人 [16] 对 PassGAN 进行了改进. 具体而言, 他们在每个口令字符的 one-hot 编码表示上添加了小幅的噪声 (通过一个超参数控制噪声的大小), 并在添加噪声后重新归一化了每个字符的分布, 以此来缓解离散特征带来的不稳定性. 在模型架构层面, 作者在生成器中采用了批归一化 (batch normalization), 从而可以更有效地使用更深的残差瓶颈块 (residual bottleneck blocks). 实验表明, 改进后的 GAN 模型在大猜测数下 ($>10^8$) 的猜测成功率显著高于原始的 PassGAN [15].

此外, Pasquini 等人 [16] 还提出了上下文 Wasserstein 自编码器模型 (context wasserstein autoencoder, CWAE) [75] 用于条件性口令猜测 (CPG), 即掩码攻击场景 (攻击者已知了用户口令的部分信息, 如长度、部分字符). 在该场景下, 攻击者可利用口令模板 (如 `**mm*91`) 来生成一系列具有相似结构的口令簇 (如 `summy91`, `tommy91` 等). 这些口令的表征向量在潜在空间 (latent space) 彼此接近, 该特性也被作者称为强局部性 (strong locality). 与之对应的是, 不同数据集的分布之间呈现出一些更为通用的特征差异, 如不同口令策略导致的不同平均口令长度和字符组成, 这使得来自同一个 (数据集) 分布的口令往往在潜在空间中更为集中, 这种特性被作者称为弱局部性 (weak locality).

在此基础上, 作者进一步提出了动态口令猜测 (dynamic password guessing, DPG) 技术. 其核心思想是将成功猜测的口令持续地用于更新潜在分布, 并模拟未知的目标口令分布. 具体而言, 给定目标口令集, 在使用生成器 (在此使用 GAN 的效果略好于 CWAE) 生成猜测的过程中, 不断统计成功猜测的口令比例, 当超过某个阈值 α 后, 使用成功猜测的口令集动态调整/更新潜在分布. 这种动态攻击策略尤其适用于跨站猜测场景, 即训练集与攻击目标来源于不同分布或分布差异较大.

随后的研究中, 研究者提出了一系列基于生成式模型的口令猜测方法. 例如, 在 2022 年, Yang 等人 [76] 提出了一种轻量级口令猜测模型 VAEPass, 该模型基于变分自编码器. 相较于 PassGAN [15], VAEPass 具有较小的参数规模和更快的模型训练速度, 同时在猜测成功率方面优于 PassGAN. 此外, Pagnotta 等人 [77] 提出了名为 PassFlow 的基于流的生成式模型用于口令猜测. 基于流的模型可以进行精确的对数似然计算和优化, 从而实现了准确的潜在变量推断. 作者通过实验证明, PassFlow 的猜测成功率优于 Pasquini 等人提出的基于 GAN 的方法.

3.3.3 基于 Seq2Seq 的口令重用模型 Pass2Path

2019 年, Pal 等人 [31] 对用户的口令重用行为进行建模, 提出了一个基于序列到序列模型 (Seq2Seq) 的口令重用模型 Pass2Path. 他们的方法首先将用户的旧口令和新口令序列分别作为模型的输入和输出, 旨在捕捉用户的口令重用模式. 然而, 这种训练方法存在一个问题: 用户的新旧口令通常非常相似, 可能包含许多相同的字符或字符串. 因此, 模型可能会学习到许多低效的 “复制” 操作, 导致其在实际应用中的泛化能力较差. 为了解决这一问题, 作者采取了新的策略, 将用户修改口令的行为视为一系列原子编辑操作. 这些原子操作包括增加、删除和替换单个位置的字符. 在此基础上, 作者将模型的输出调整为从旧口令到新口令的编辑操作序列. 举例来说, 如果从口令 `wang123` 变换为 `wang1!`, 对应的原子操作序列为 [(在索引为 5 的位置删除字符 2), (在索引为 6 的位置删除字符 3), (在索引为 7 的位置插入字符!), 结束], 模型的输入为原始口令 `wang123` 的字符序列, 输出为对应的原子操作序列. 这一新的建模方式极大地提升了模型的猜测成功率. 图 6 为 Pass2Path 模型的示意图.

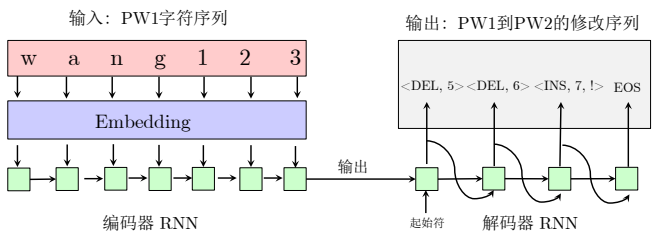


图 6 基于编辑操作序列的口令重用模型 Pass2Path

Figure 6 Password reuse model Pass2Path based on editing operation sequence

3.3.4 基于多步决策机制的口令重用模型 Pass2Edit

2023 年, Wang 等人 [35] 指出如果将用户的口令修改过程建模为一系列原子修改操作 (例如, 删除或插入特定的口令字符), 并使用神经网络来预测修改操作序列, 那么每个修改操作可能对后续的修改操作产生一定的影响. 然而, 现有方法 (如 3.3.3 节所介绍的 Pass2Path [31]) 无法准确学习到单个修改操作对当前口令的作用效果. 为了更准确地表示用户的口令修改/重用行为, Wang 等人首次将口令重用攻击建模为一个多分类问题, 并提出了一个名为 Pass2Edit 的生成模型. 该模型采用多步决策机制, 能够在口令经历复杂变换时学习到修改操作对当前口令的作用效果.

具体而言, Pass2Edit 模型的输入由原始口令和之前单个修改步骤所产生的当前口令共同组成. 模型的输出是下一步的原子修改操作. 在输出单步修改操作后, 模型将该修改操作应用到输入端的当前口令中, 作为下一步的模型输入. 如此循环迭代, 直至输出终止操作. 通过这种多步决策学习机制, 模型能够学习到口令修改过程中每个修改操作对原始口令的影响. 例如, 假设原始口令为 `wang123`, 目标口令为 `wang1!`, 那么这两个口令之间的编辑操作序列为 [(在索引为 5 的位置删除字符 2), (在索引为 6 的位置删除字符 3), (在索引为 7 的位置插入字符!), 结束]. 神经网络的输入涵盖了原始口令以及当前已修改的口令, 而输出则是下一步的单步编辑操作. 首先, 将两个原始口令作为模型的输入, 第一步输出为 (在索引为 5 的位置删除字符 2); 然后, 将该操作应用于原始口令 `wang123`, 得到当前修改后的口令 `wang13`, 并将其与原始口令一同作为下一步的模型输入, 输出为 (在索引为 6 的位置删除字符 3); 再依照同样的方式, 循环调用该网络, 直至遇到结束操作. 图 7 为该方法的一个示例说明.

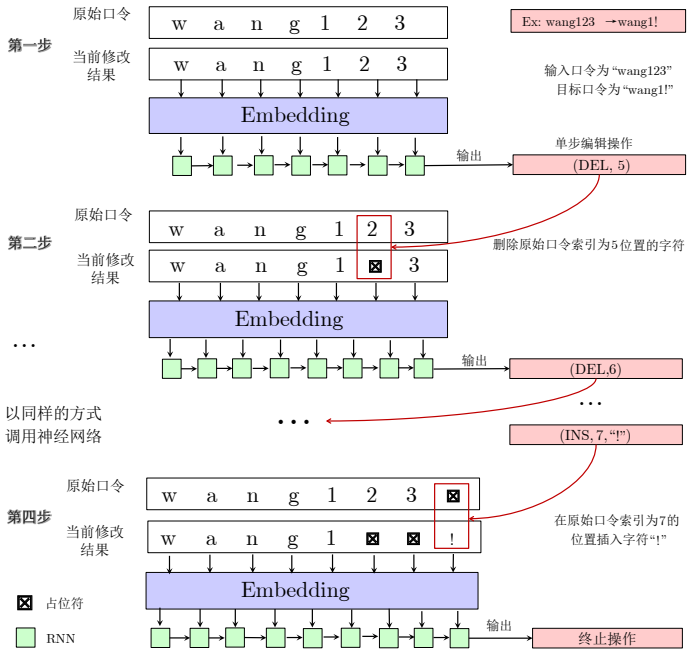


图 7 基于多步决策机制的口令重用模型 Pass2Edit [35]

Figure 7 Password reuse model Pass2Edit based on multi-step decision-making mechanism [35]

3.3.5 基于规则的自适应攻击算法 ADaMs

基于规则的字典攻击通常由两个关键组成部分构成: 一个包含 n 个口令的基础字典集及一个包含 m 个变换规则的规则集. 在生成猜测时, 会将规则集中的每个规则变换应用于基础字典中的每个口令, 从而生成 $n \times m$ 个口令作为猜测字典. 然而, 这种方法会导致大量无效或重复的猜测, 因为在现实世界中, 不同基础口令与不同规则的适配程度存在巨大的差异. 举例来说, 长度较短的口令相对于长度较长的口令更适于被应用在尾部添加字符 (串) 的规则. 类似地, 对于由纯小写字母构成的口令, 更适于应用全体字符大写的规则, 而对于由字母和数字混合构成的口令, 这种规则的适用性可能较低.

针对此问题, Pasquini 等人 [24] 在 2021 年提出了一种新型字典攻击技术, 该技术无需监督或专业领域知识, 能够自动模拟真实攻击者采用的高级猜测策略, 在应对不完善的字典攻击配置方面表现出更强的适应性. 具体而言, 对于给定基础字典 W 和规则集 R, 通过引入一个足够大的字典 X, 为基础字典中的每个口令 w_i 和规则集中的每个规则 r_j 分配一个类别标签: 标签为 1 表示 $r_j(w_i) \in X$, 0 表示 $r_j(w_i) \notin X$. 通过这种方式, 每个口令将被分配 $|R|$ 个标签 (即规则的数量), 此时, 基础口令与规则的适配程度可视为

一个多标签分类问题 (multi-label classification). 随后, 作者使用深度神经网络来学习规则集所对应的适配函数. 模型训练完成后, 模型将对每个口令 w_i 和每个规则 r_j 输出一个位于 0 到 1 之间的适配分数, 其中接近 1 的数值表示 r_j 对 w_i 是适配的. 相反, 输出值接近 0 表示真实情况下将 r_j 应用于 w_i 的概率极低. 最后, 通过设置适配分数阈值来排除那些对给定口令无效的规则 (例如, 设置阈值为 0.5 即可排除 85%/90% 的无效规则 [24]), 从而大大提升了原始字典攻击的攻击效率. 在此基础上, 他们引入了动态猜测策略, 即将每一个新猜出的口令都插入到原始基础字典中, 这在现实情况下对应了专家根据目标情况即时调整猜测策略的能力, 从而进一步提升了攻击效率.

3.3.6 Chunk-Level 口令猜测框架

2021 年, Xu 等人 [33] 指出现有口令猜测模型大多将口令视为连续的字符序列, 或假设不同类型字段之间相互独立, 这些方式均无法捕获口令中存在的独有模式, 如 `4ever`、`ing` 等. 针对此问题, Xu 等人通过引入字节对编码 (byte-pair-encoding, BPE) 算法, 将口令划分为更细粒度的块级 (chunk-level) 表征. 例如, 口令 `p@ssw0rd4ever` 可以通过 BPE 划分为 `p@ssw0rd` 和 `4ever` 两块. 使用 BPE 算法将口令切分成 chunk 的大体流程如图 8 所示.

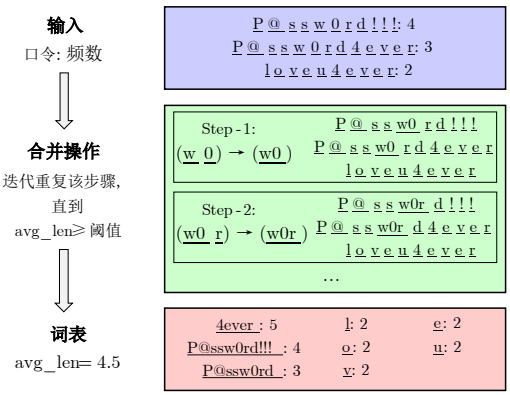


图 8 使用 BPE 算法将口令切分成 chunk 表征 [33]

Figure 8 Use BPE algorithm to split password into chunk representation [33]

在此基础上, Xu 等人 [33] 构建了三个基于 chunk 划分的口令猜测模型: Chunk-level Backoff、Chunk-level PCFG 和 Chunk-level FLA. 对于 Chunk-level Backoff, 它与普通 Backoff [22] 的区别在于使用 chunk 代替原始的单个字符作为计算转移概率的最小单元; 此外, Chunk-level Backoff 还限制了预测下个 chunk 所需考虑的最大 chunk 前缀数目. 其训练与生成过程与普通 Backoff 类似. 对于 Chunk-level PCFG, 它在原有 PCFG [21] 大/小写字母 U/L、数字 D 和特殊符号 S 的基础上, 额外引入了 DM、TM 和 FM 三种类型的标签, 分别代表两种类型字符混合的字符串 (double mixed type, 如 `4ever` 可表示为 DM_5)、三种类型字符混合的字符串 (three mixed type, 如 `p@ssw0rd` 可表示为 TM_8) 和四种类型字符混合的字符串 (four mixed type, 如 `P@$$w0RD` 可表示为 FM_8). 其训练与生成过程也与普通 PCFG 类似. 对于 chunk-level FLA, 它与普通 FLA [14] 的区别在于使用 chunk 代替原始的单个字符, 模型通过先前的 chunks 作为上下文环境来预测下个 chunk 出现的概率. 由于 chunk 的种类远多于单个字符的种类, 作者使用了一个嵌入层来代替原始字符级 FLA 所使用的独热编码层 (one-hot encoding layer). 其余过程和普通 FLA 类似. 作者通过实验证实, 三个 chunk-level 模型的猜测成功率均优于对应的原始模型.

3.3.7 基于随机森林的口令猜测框架 RFGuess

2023 年, Wang 等人 [26] 以随机森林算法为例, 通过对口令字符的重新编码, 设计了基于经典机器学习技术的新型口令猜测框架 RFGuess, 涵盖漫步猜测场景和两个定向猜测场景 (即, 基于个人信息的定向猜测和基于口令重用的定向猜测).

首先介绍漫步口令猜测算法 RFGuess. 给定一个口令 (如 `password`), 将该口令每个字符前面的 n 阶字符串作为分类目标 (如 `passwo`), 而将该字符 (如 `passwo` 所对应的字符 r) 作为对应待分类字符串的正

确分类标签. 其中, n 阶字符串中的每个字符使用 [字符类型 (即, 数字, 小写/大写字母, 特殊符号), 字符所在类型序号 (如, 字母 `a` 位于 `a`–`z` 序列的第一个), 字符所在键盘行号, 字符所在键盘列号] 四个维度来表示, n 阶字符串整体再使用 [已遍历的字符长度, 当前字符所在的相同字符类型的段已经遍历长度] 两个维度来表示, 共计 $4n+2$ 个维度. 下面给出一个具体示例.

以口令 `qwer654321` 为例, 取其 6 位前缀 `wer654`. 首先, 可以将 `wer654` 中的每个字符用一个 4 维特征向量进行唯一表示. 例如, 字符 r 被表示为 [3, 18, 2, 4], 其中 3 表示小写字母, 18 是 r 在小写字母序列 a–z 中的排名, 2 和 4 分别是 r 在键盘上的行和列位置. 如此, `wer654` 可以由一个 24 维特征向量表示 ($24 = 4 \times 6$). 然后, 添加一个 2 维向量 [7, 3] 来表示前缀 `wer654` 的长度特征, 其中 7 表示数字 4 是 `qwer654321` 的第 7 个字符, 3 表示数字 4 是数字段 `654321` 的第 3 个字符). 在训练阶段, 将训练集中的所有口令转化为 n 阶字符串的形式. 模型的输入是每个 n 阶字符串的向量表示, 而输出则是与该字符串对应的字符标签. 生成猜测的过程类似于 Markov 模型. 从一个 n 阶起始字符串开始, 模型逐个字符地生成, 直到遇到终止符. 该过程可视为模型通过观察之前的字符来预测下一个字符, 以此生成可能的口令猜测. 图 9 为使用决策树对口令前缀进行分类的简单示例.

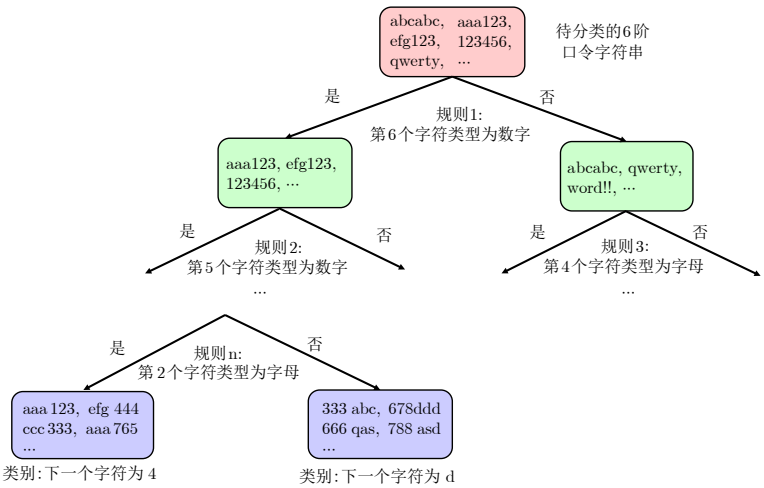


图 9 使用决策树对 n 阶口令字符串进行分类 [26]

Figure 9 Classify n-order password strings using decision trees [26]

基于个人信息的定向口令猜测算法 RFGuess-PII 与漫步算法 RFGuess 原理相同, 不同之处在于对包含个人信息口令片段的编码方式不同. 具体来说, 对于定向算法, 个人信息的使用行为除了没有键盘特征外, 其他与普通字符相似. 首先, 个人信息也有不同类型之分 (如邮箱、用户名、姓名等, 这类比于普通字符的字符类型), 而且不同类型个人信息之间的使用行为也互不相关. 其次, 给定个人信息类型, 其使用行为也有着一定的规律/关联性 (如姓名包含姓名全称、姓名缩写、姓氏等一系列变型, 这可类比于普通字符的字符序号). 因此, 使用 [个人信息类型, 个人信息行为序号, 0, 0] 这四个维度表示定向猜测算法中包含个人信息的片段特征, 其中, 最后两个 0 是为了与普通字符的特征对齐. 以口令 `wang123` 为例, 个人信息片段 `wang` 可表示为 (10, 1001, 0, 0), 其中 10 代表姓名这一个人信息种类, 1001 代表使用姓名信息的第一种变型, 即姓氏. 该算法的训练和猜测生成过程也与 RFGuess 相似.

基于口令重用的定向口令猜测算法 RFGuess-Reuse 可分为两个主要阶段: 结构级变换和字段级变换. 在结构级变换阶段, 统计训练集中所有口令对在结构层面的转换操作及对应频率. 例如, 统计从结构 $\mathsf{L}_8\mathsf{S}_2$ (表示长度为 8 的字母片段和长度为 2 的特殊字符片段) 变化为结构 $\mathsf{L}_7\mathsf{D}_3$ (表示长度为 7 的字母片段和长度为 3 的数字片段) 的口令对的频率, 并依据概率进行排序. 接下来, 在字段级变换阶段, 使用随机森林算法对字段级转换操作进行预测. 例如, 在字母片段中, 若将 `password` 转换为 `passwor`, 那么随机森林的作用是输入 `password`, 然后预测应该采取的编辑操作, 即删除最后一个字符 `d`. 以下给出一个完整的示例. 给定一个口令 `password!!`, 生成目标口令 `p@sswor123` 的概

率计算过程为 Pr(password!! → p@sswor123) = Pr(password!! → password) × Pr(password → passwor) × Pr(passwor → passwor123) × Pr(passwor123 → p@sswor123) × p_4, 其中, Pr(password!! → password) (即 $\mathtt{L}_8\mathtt{S}_2 \to \mathtt{L}_8$) 和 Pr(passwor → passwor123) (即 $\mathtt{L}_7 \to \mathtt{L}_7\mathtt{D}_3$) 表示结构级转换的概率, 可以通过训练集中口令对的统计数据获得; Pr(password → passwor)(即删除单个字符 d) 和 Pr(passwor123 → p@sswor123)(即 a → @) 是字段级转换的概率, 可以通过已训练的随机森林模型获得; p_4 是经过四次转换后结束的概率, 同样可以通过训练集的统计数据获得.

3.3.8 基于双向 Transformer 的口令猜测框架 PassBERT

2023 年, Xu 等人 [32] 将自然语言处理领域的预训练/微调技术用于口令猜测, 提出了一个基于双向 Transformer 的口令猜测框架 PassBERT. 该框架涵盖三种不同类型的攻击场景: (1) 基于部分口令信息的掩码攻击场景 (即攻击者已知了用户口令的部分信息, 如长度和/部分字符); (2) 基于口令重用的定向攻击场景; (3) 基于规则的自适应攻击场景. 具体来说, 作者首先使用大规模口令数据集 (Rockyou-2021), 基于掩码语言建模 (masked language modeling), 建立了口令预训练模型. 其中, 掩码语言建模是指, 训练模型使其能利用包含掩码字符的口令模板 (如 il*v*u4*ve*) 来恢复出该模板所对应的原始完整口令 (即 iloveu4ever). 在此之后, 作者根据三种攻击场景的不同特点, 对预训练模型进行了针对性微调, 即为三种攻击场景设置了不同的监督信号 (supervision signals).

具体来说, 掩码攻击模型的输入为口令模板 (与预训练的不同之处在于, 将原始 BERT 模型的 15% 字符掩盖比例提升为 50%), 输出为完整的原始口令; 定向攻击模型的输入为旧口令的字符序列, 输出为从旧口令到新口令的最短编辑操作序列 (作者定义了三种不同的操作, 分别为保持不变、删除和单个/双个字符替换); 自适应攻击模型的输入为口令字符序列, 输出为针对所选规则集中的每个规则与该口令匹配程度的一个数值 (位于 0 到 1 之间). PassBERT 的整体框架如图 10 所示. 作者通过实验证实, 所提出的三个模型的猜测成功率均优于对应场景下最先进的口令模型.

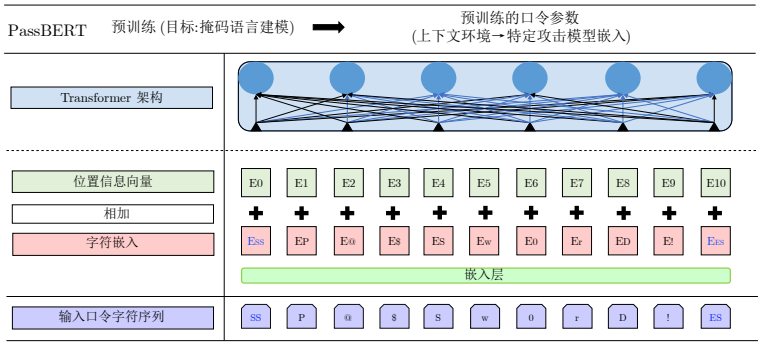


图 10 PassBERT 的整体框架 [32]

Figure 10 Overview of PassBERT framework [32]

4 口令猜测算法评估

4.1 口令猜测算法评价指标

研究口令猜测算法, 一个首要问题是如何恰当地评测不同算法的猜测能力. 在一些早期研究 (如文献 [17, 65]) 中, 主要通过 "猜测成功率 vs. 猜测时间" 来衡量特定口令的抗猜测能力. 这一指标严重依赖于攻击者自身的硬件计算能力 (如是否利用了 GPU) 和算法的具体实现方式 (如实现语言、口令生成过程是否采用了剪枝优化等), 往往难以进行公平的比较. 2009 年, Weir 等人 [21] 开始采用 "猜测成功率 vs. 猜测次数 (已搜索空间)" 这一评价指标, 即通过对比不同算法在相同猜测数下, 可猜出的口令比例来衡量, 即 猜测成功率 $= \frac{\text{成功猜测的用户数 (在前}N\text{次猜测内)}}{\text{测试集中的用户总数}} \times 100\%$.

例如, 给定 10 000 条测试口令, 算法 A 在 100 万猜测数下, 成功猜测出其中的 4 000 条口令, 则算法 A 在 100 万猜测数下的猜测成功率为 40% ($=\frac{4000}{10000} \times 100\%$). 2010 年, Dell'Amico 等人 [25] 指出, 这一

指标只依赖于猜测算法自身的能力和用户选择口令的方式, 既与口令存储系统的独特配置环境无关 (如采用了什么 hash 函数、是否加盐), 也与攻击者的硬件计算能力和算法实现方式无关. 例如, 攻击者可利用算法/工具提前生成猜测集, 再去发动猜测攻击. 这样, "猜测成功率 vs. 猜测次数" 这一评价指标就实现了对算法猜测能力的公平对比, 学术界逐渐接受了这一指标.

在现实世界中, 网站或服务器为了防御口令猜测攻击, 会采用加盐 (对抗预计算技术, 如彩虹表)、增加 Hash 次数、使用内存困难函数 (MHF) 等安全防御措施, 这会大大提高攻击者单次猜测的成本, 而该成本与系统环境和攻击者的硬件计算能力息息相关. 因此, 在评估猜测算法的实用性时, 算法的计算效率理应成为猜测算法的评价指标之一. 具体来说, 算法的计算效率主要体现在训练时间、运行所需要的硬件支持 (是否需要 GPU、占用内存大小等)、模型大小以及口令生成速度. 此外, 尽管不断泄露的口令文件为攻击者提供了丰富的资源, 但仍存在大量训练数据资源匮乏的攻击场景 (如银行和除了中、美、俄以外的其它国家/地区). 因此, 算法充分发挥性能所需要的训练集大小也应纳入考虑范围.

4.2 口令猜测场景

口令猜测场景可以根据不同的标准进行分类, 包括是否使用目标的个人信息 (如姓名、生日、邮箱、用户名等)、是否与服务器交互、允许的猜测次数的大小等.

根据是否使用目标的个人信息, 口令猜测攻击可分为漫步猜测与定向猜测. 所谓漫步猜测, 是指攻击者不关心攻击的具体目标, 其目的是破解出尽可能多的口令; 所谓定向猜测, 是指攻击者利用与目标相关的各种信息 (如姓名、邮箱、已泄露的旧口令等), 其目的是尽快破解出指定目标的口令. 根据攻击者所拥有目标信息的不同, 本文将定向猜测场景进一步分为基于个人信息 (如姓名、邮箱、生日、用户名等个人信息) 的定向猜测和基于口令重用的定向猜测. 在定向猜测场景下, 指标猜测成功率的计算方法如下: 为测试集中的每个用户生成一定数量的猜测集, 然后统计测试集中所有用户在其对应猜测集下的破解情况. 例如, 定向猜测算法 A 在 100 猜测数下猜测成功率为 20% 表示: 对于测试集中的所有用户, 有 20% 用户的口令在其对应猜测集的前 100 次猜测内被成功破解.

根据是否与服务器交互, 口令猜测攻击可分为在线猜测与离线猜测. 在线猜测往往难以避免: 任何拥有浏览器的人都可以对公开界面的服务器发起基本的在线猜测攻击. 为了抵御在线攻击, 服务器往往被建议部署登录速率限制机制 (rate-limiting), 如账户锁定 (account lockout) 和登录节流 (login throttling). 然而, 现实情况却不尽人意. Lu 等人 [78] 在 2018 年的研究表明, 约 72% 的网站 (131/182) 允许频繁的、不成功的登录尝试, 而没有账户锁定或登录节流措施. 对于这些网站, 攻击者每天可以进行超过 85 次的登录尝试, 这远超过 NIST-800-63-3 所建议的 30 天内每个账户允许的错误口令登录次数不应超过 100 [79]. 事实上, 即使是采取了严格速率限制机制的网站, 其每月允许的错误口令登录次数也远超过 100. 例如, 亚马逊网站的速率限制机制为每小时允许连续五次错误登录尝试, 据此推算, 攻击者每天可至多进行 120 次登录尝试, 每个月可进行 3600 次登录尝试. 与之对应的是, 在现有研究中 (如文献 [26, 28, 56, 80, 81]), 在线猜测算法的最大猜测数通常设置为 10^3~10^4.

离线猜测需要攻击者能够获取服务器泄露的口令文件, 可尝试的猜测次数仅仅受限于攻击者本身的计算资源和愿意投入的成本. Florêncio 等人 [80] 指出, 离线猜测攻击只在非常受限的情形下才构成现实威胁: 服务器端采用了正确的口令加盐 hash 方法, 口令文件被泄露且未被察觉. 对于猜测算法的评估来说, 在线猜测与离线猜测的本质区别在于猜测次数的大小不同. Florêncio 等人 [80] 通过分析指出, 10^{14} 可以作为离线猜测的一个下界. 然而, 现有研究中 (如文献 [16, 21, 22, 52]), 受计算资源的限制, 几乎所有的猜测算法在被评估时, 实际生成猜测数的最大数量级为 10^{12}(见文献 [24] 的图 8), 一般数量级在 10^7~10^{10} [16, 21–23, 52], 尚未见实际生成超过 10^{13} 的猜测算法研究.

4.3 蒙特卡洛采样方法

对于算法在大猜测数的表现, 通常采用蒙特卡洛采样方法进行评估 [30]. 蒙特卡洛采样的基本原理是, 基于猜测模型所定义的口令分布随机抽样, 通过样本口令的概率和测试集中每个口令在猜测模型下的概率, 估计测试集中每个口令的强度 (即, 被攻破所需的猜测数), 最终得到该模型在对应测试集下的猜测成功率曲线. 该方法只适用于概率模型 (即猜测模型所对应的口令空间的概率加和为 1.0).

以 Markov 模型 [22] 为例 (记为 p), 假设我们的字符表包括 95 个可打印字符和一个终止符. 在训练完成后, 模型逐个字符生成样本口令, 具体流程为: 每个字符都被模型 p 分配一个概率值, 表示该字符作为

生成口令首个字符的概率; 将 0 到 1 这个区间划分成 96 个小区间, 每个区间代表一个字符以及该字符所对应的概率范围; 在生成口令时, 使用一个随机种子生成一个 0 到 1 之间的随机概率, 然后确定它位于这 96 个小区间中的哪一个, 从而选择相应的字符作为模型生成口令的首个字符, 记为 c_1; 接下来以同样的方式生成该口令的第二个字符, 此时, 字符表中的每个字符会被模型分配一个新的概率, 表示该字符出现在 c_1 之后的概率; 重复该抽样过程, 将抽取的字符依次添加至当前字符串并记录其对应的随机数, 直到抽取到终止字符为止. 这样就完成了一个口令样本的采样, 而其采样概率可以通过将所有采样字符对应的随机数相乘来得到. 重复此过程, 生成大小为 n 的样本集, 记作 Θ, 每个生成的口令样本记为 β, 其对应的生成概率用 $p(\beta)$ 表示. 随后, 计算测试集中的每个口令 (记为 α) 在模型 p 下的概率, 记为 $p(\alpha)$. 再依据以下公式对口令 α 的猜测数进行估算:

$$C_{\Delta} = \sum_{\beta \in \Theta} \begin{cases} \frac{1}{n \cdot p(\beta)} & \text{if } p(\beta) > p(\alpha) \\ 0 & \text{otherwise} \end{cases}. \tag{1}$$

Dell'Amico 等人 [30] 证明 C_{Δ} 的期望即为口令 α 的预估猜测数; 随着样本大小 n 的增加, 其方差减小, 收敛速度为 $\mathcal{O}(1/\sqrt{n})$. 在实现过程中, 只需将样本口令对应的概率存储在数组 $A = [p(\beta_1), p(\beta_2), \cdots, p(\beta_n)]$ 中, 依概率降序排列, 同时创建一个数组 C, 使得 C 的第 i 个元素为 $C[i] = \frac{1}{n}\sum_{j=1}^{i} \frac{1}{A[j]} = C[i-1] + \frac{1}{nA[i]}$, 概率为 $A[i]$ 的口令所对应的猜测数为 $C[i]$. 在计算口令猜测数时, 通过二分查找, 查找 A 中最右边的 $A[j]$ 值对应的索引 j, 使得 $A[j] > p(\alpha)$; C_{Δ} 的输出即为 $C[j]$. 由于使用了二分查找, 该过程的时间复杂度为 $\mathcal{O}(\log n)$.

2023 年, Liu 等人 [82] 注意到 Dell'Amico 等人的蒙特卡洛估计方法没有提供关于猜测数范围的统计置信区间, 也明确指出存在高度不确定性的情况. 基于这一发现, 他们提出了自信蒙特卡洛评估, 该方法在给定置信度时, 能够提供猜测曲线 (即猜测成功率 vs. 猜测数) 的上下界, 进一步增强了原始蒙特卡洛估计方法的科学性和完整性. 在第 4.4 节中, 将从实验的角度证实蒙特卡洛估计方法的有效性.

4.4 实验设置与算法评估

科学的实验设置对于公平评估口令猜测算法的表现至关重要. 如本节所示, 不同算法通常在不同的猜测场景中表现出优势. 以往的研究表明, FLA 算法在大规模猜测数 (如 $>10^{10}$) 的情况下相对于 PCFG 和 Markov 算法表现出了明显的优势 [14], 而 RFGuess 算法则在小规模训练集 (如 1 万训练集) 和同站测试场景下优于其他方法 [26]. 为了确保对不同算法的评估具有准确性、公平性和合理性, 本文首先设计了一系列通用猜测场景, 以观察各算法在不同条件下的猜测成功率. 在此基础上, 考虑到不同算法的潜在优势, 根据各算法的原始文献, 设置了特定的实验场景, 以展示它们在自身擅长领域的性能. 这种兼顾通用性和定制化的实验场景设计确保了对各算法能力的全面评估, 有助于更好地理解它们的应用范围和局限性.

4.4.1 漫步攻击场景

漫步猜测场景的实验设置如表 9 所示. 在场景 #1∼#6 中, 分别选取 50%、30% 和 10% 的 Dodonew/000Webhost 数据作为训练集, 以研究模型对训练集规模的敏感性. 对于测试集, 首先, 使用 20% 的 Dodonew/000Webhost 数据创建基础测试集, 模拟同站猜测场景 (场景 #10 和场景 #15). 接着, 从这 20% 的 Dodonew/000Webhost 数据中随机抽取不同规模的子测试集 (1 万、10 万和 100 万个口令, 对应场景 #7∼#9 和场景 #13∼#15), 以观察测试集规模对实验结果的影响. 同时, 排除 20% Dodonew/000Webhost 中在训练集中已经出现过的口令, 以评估模型生成新口令的能力, 即泛化能力 (对应场景 #11 和场景 #17). 最后, 使用随机选取的 100 万条 Taobao 和 Wishbone 数据集作为测试集, 以模拟跨站猜测场景 (场景 #12 和场景 #18). 值得注意的是, 000Webhost 要求口令长度大于 6, 且至少包含一个数字和字母, 因此还进一步对数据集 Wishbone 按照 000Webhost 的口令策略进行过滤, 以观察口令策略对实验结果的影响 (场景 #19).

本文对比了几种具有代表性的口令猜测方法, 包括 PCFG/PCFGv4.1 [21]、Markov [22]、Backoff [22]、FLA [14]、DPG [16]、RFGuess [26]、ADaMs [24] 和 PassBERT(-ARPG) [32]. 模型参数遵循原始论文的最佳推荐. 针对概率模型 (PCFG、Markov、Backoff、FLA 和 RFGuess), 根据各自的训练/生成速度和内存消耗 (见表 10), 实际生成了不同数量级的猜测数, 并与相应的蒙特卡洛评估结果进行了对比 (场景 #12 和 #18). 对于 DPG, 使用 80% 000Webhost/Dodonew 作为训练集, 以剩下 20% 数据集作为攻击目标,

生成了约 10^9 个猜测. 而对于 ADaMs 和 PassBERT(-ARPG), 其训练过程通常需要一个基础字典、一个规则集和一个大规模目标集, 本文采用作者提供的预训练模型和规则, 将 80% 000Webhost/Dodonew 用作基础字典, 实际生成了约 10^9 个猜测. 详细情况如表 9 所示.

图 11 (a)、图 11 (b)、图 11 (d) 和图 11 (e) 为所有方法的对比结果. 整体看来, 在不去除训练集与测试集交叉部分的同站猜测场景中 (即图 11 (a) 和图 11 (d)), 概率模型 (如 Backoff [22]、FLA [14] 和 RFGuess [26]) 的猜测成功率通常优于基于规则的方法 (如 ADaMs [24] 和 ARPG [32]. 这里需要注意的是, 图 11 (a)、图 11 (b)、图 11 (d) 和图 11 (e) 中的 "_pwdpro" 和 "_generated" 分别代表使用两种不同的规则集所训练的模型), 这些结果说明概率模型能更有效地拟合训练集口令分布. 值得注意的是, 从 10^4 猜测数开始, DPG [16] 在英文数据集 (见图 11 (d)) 的猜测成功率逐渐优于其它方法. 原因在于 DPG 在该猜测数下, 启动了动态调整机制, 将成功猜测的口令用于调整/更新潜在分布. 正因如此, 在去除测试集中在训练集中出现过的口令后 (即图 11 (b) 和图 11 (e)), DPG [16] 表现尤为出色. 而其余各个方法的猜测成功率均显著下降, 在 10 万猜测数内, 猜测成功率几乎为 0, 且彼此相差不大. 对于字符级自回归模型 (即 Markov [22]、Backoff [22]、FLA [14] 和 RFGuess [26]), Backoff、FLA 和 RFGuess 由于在不同程度上缓解了 Markov 模型存在的数据稀疏和过拟合问题, 猜测成功率通常大于等于 Markov.

表 **9** 漫步猜测场景实验设置

Table 9 Experimental setup of trawling guessing scenarios

#	训练集	测试集	猜测算法及其对应的最大猜测数: 实际生成 vs. 蒙特卡洛模拟								
			PCFG [21]	PCFG v4.1 [21]	Markov [22]	Backoff [22]	FLA [14]	RFGuess [26]	DPG [16]	PassBERT (-ARPG) [32]	AdaMs [24]
1	50% Dodonew	Dodonew(10^5)	– 10^{14}	–	– 10^{14}	– 10^{14}	– 10^{14}	– 10^{14}	–	–	–
2	30% Dodonew		– 10^{14}	–	– 10^{14}	– 10^{14}	– 10^{14}	– 10^{14}	–	–	–
3	10% Dodonew		– 10^{14}	–	– 10^{14}	– 10^{14}	– 10^{14}	– 10^{14}	–	–	–
4	50% 000Webhost	000Webhost(10^5)	– 10^{14}	–	– 10^{14}	– 10^{14}	– 10^{14}	– 10^{14}	–	–	–
5	30% 000Webhost		– 10^{14}	–	– 10^{14}	– 10^{14}	– 10^{14}	– 10^{14}	–	–	–
6	10% 000Webhost		– 10^{14}	–	– 10^{14}	– 10^{14}	– 10^{14}	– 10^{14}	–	–	–
7	80% Dodonew	Dodonew(10^4)	10^9 –	10^9	10^8 –	10^8 –	10^8 –	10^7 –	–	–	–
8		Dodonew(10^5)	10^9 –	10^9	10^8 –	10^8 –	10^8 –	10^7 –	–	–	–
9		Dodonew(10^6)	10^9 –	10^9	10^8 –	10^8 –	10^8 –	10^7 –	–	–	–
10		20% Dodonew	10^9 –	10^9	10^8 –	10^8 –	10^7 –	10^7 –	10^9	10^9	10^9
11		20% Dodonew(∗)†	10^9 –	10^9	10^8 –	10^8 –	10^8 –	10^7 –	–	–	–
12		Taobao(10^6)	10^9 10^{14}	–	10^8 10^{14}	10^8 10^{14}	10^8 10^{14}	– –	–	–	–
13	80% 000Webhost	000Webhost(10^4)	10^9 –	10^9	10^8 –	10^8 –	10^8 –	10^7 –	–	–	–
14		000Webhost(10^5)	10^9 –	10^9	10^8 –	10^8 –	10^8 –	10^7 –	–	–	–
15		000Webhost(10^6)	10^9 –	10^9	10^8 –	10^8 –	10^8 –	10^7 –	–	–	–
16		20% 000Webhost	10^9 –	10^9	10^8 –	10^8 –	10^8 –	10^7 –	10^9	10^9	10^9
17		20% 000Webhost(∗)†	10^9 –	10^9	10^8 –	10^8 –	10^8 –	10^7 –	–	–	–
18		Wishbone(10^6)	10^9 10^{14}	10^9	10^8 10^{14}	10^8 10^{14}	10^8 10^{14}	10^7 10^{14}	–	–	–
19		Wishbone(LD len≥6)	10^9 –	10^9	10^8 –	10^8 –	10^8 –	10^7 –	–	–	–

† ∗ 表示去除了测试集中在训练集中出现过的口令, 即 20% Dodonew 数据集中不包含出现在 80% Dodonew 数据集中的口令.

表 **10** 不同漫步猜测模型的计算效率 †

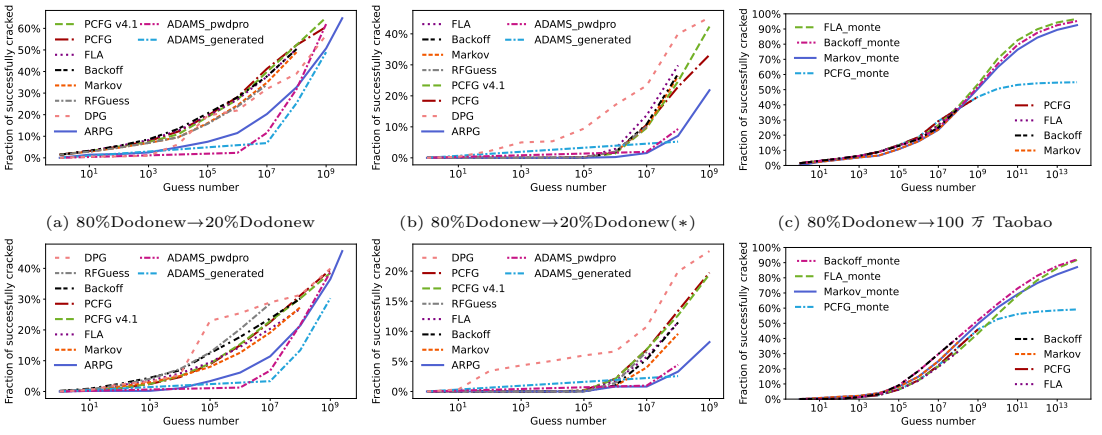
Table 10 Performance of different trawling guessing models

模型	训练时间	模型大小	口令生成速度 (PW/s)
RFGuess [26]	0.3 h	4.5 G‡	130
PCFG [21]	24 s	93.2 M	82 372
3-order Markov [22]	102 s	1.4 G	13 303
FLA [14]	16 h	5.8 M	2500
DPG [16]	2.5 h	1.6 M	3175

† CPU: Xeon silver 4210R 2.4GHz; GPU: GeForce RTX 3080 (使用 500 万 CSDN 数据集).
‡ 此大小表示模型在压缩后的尺寸, 通过将 joblib 中的压缩参数设置为 3, 原始模型大小约为 40 GB.

表 10 列出了几个主要模型的训练时间、模型大小和口令生成速度. 各模型在不同场景的运行特性决定了各自的适用性. 例如, 训练时间相对较短的模型 (如 RFGuess [26]) 更适用于在线猜测场景; 模型大小

较小的模型 (如 FLA [14]) 更适合用于客户端层面的口令强度评估 (client-side PSM); 而生成速度较快的模型 (如 PCFG [21]) 更适用于大规模离线口令猜测任务. 值得注意的是, 在不断涌现各种口令猜测算法的今日, 综合考虑模型的训练/生成速度和猜测成功率, PCFG [21] 仍表现卓越.



(a) 80%Dodonew→20%Dodonew (b) 80%Dodonew→20%Dodonew(∗) (c) 80%Dodonew→100 万 Taobao

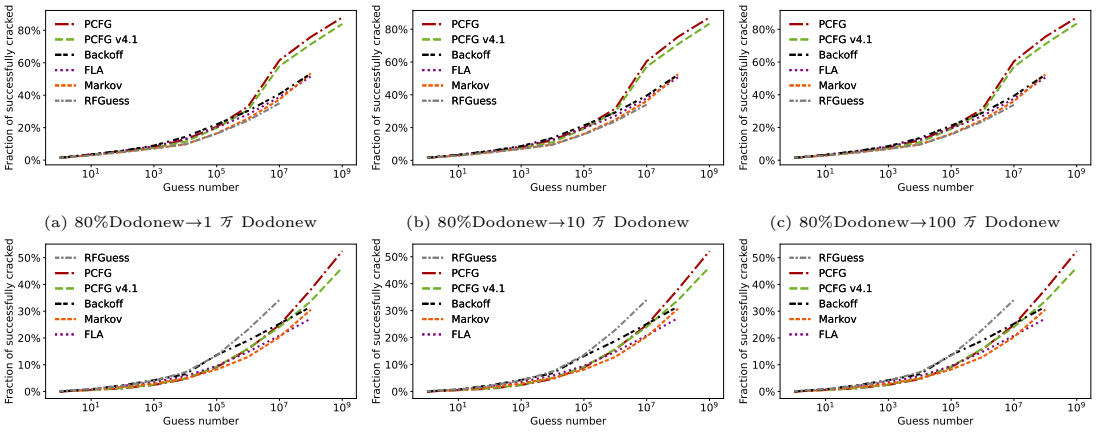
(d) 80%000Webhost→20%000Webhost (e) 80%000Webhost→20%000Webhost(∗) (f) 80%000Webhost→100 万 Wishbone

图 11 漫步猜测实验结果图 (∗ 表示去除了测试集中在训练集中出现过的口令)

Figure 11 Trawling guessing scenarios (∗ means removing passwords seen in training set from test set)

图 11 (c) 和图 11 (f) 为 PCFG [21]、Markov [22]、Backoff [22] 和 FLA [14] 利用实际生成的猜测集与使用蒙特卡洛估计 [30] 所得到的 "猜测数 vs. 猜测成功率" 曲线. 观察发现, 每种方法对应的两条曲线几乎重合. 这从实验的角度证明了蒙特卡洛估计方法在给定猜测数下的准确性.

不同方法在不同测试集大小下的实验结果如图 12 所示. 观察发现, 测试集大小从 1 万变化至 >100 万时, 各个方法的结果几乎一致 (猜测成功率误差 <1%). 这表明, 一旦测试集大小超过 1 万, 不同方法的猜测成功率就会趋于稳定, 这与文献 [30] 中推荐的测试集大小一致.



(a) 80%Dodonew→1 万 Dodonew (b) 80%Dodonew→10 万 Dodonew (c) 80%Dodonew→100 万 Dodonew

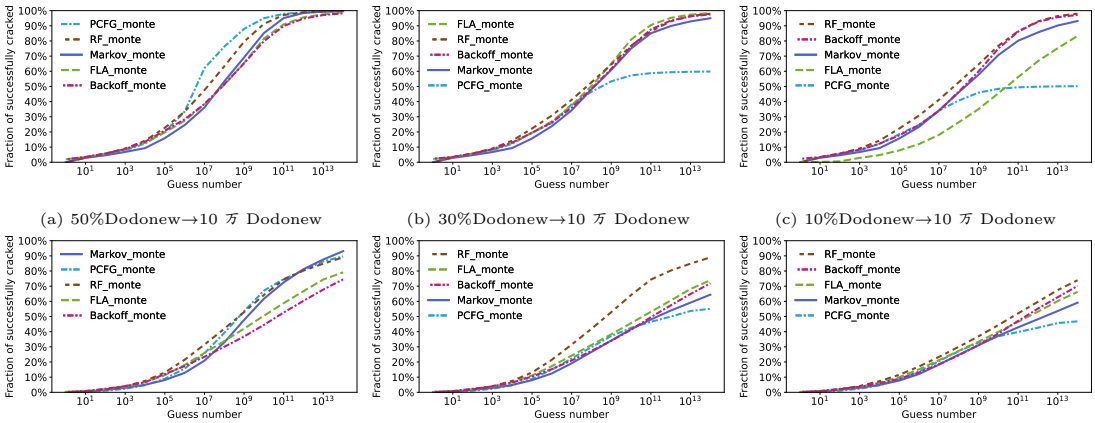
(d) 80%000Webhost→1 万 000Webhost (e) 80%000Webhost→10 万 000Webhost (f) 80%000Webhost→100 万 000Webhost

图 12 漫步猜测实验结果图 (测试集大小对实验结果的影响)

Figure 12 Experimental results of trawling guessing (Impact of testing set size on experimental results)

不同方法使用不同大小训练集的实验结果如图 13 所示. 观察发现, RFGuess [26] 在训练集较小的情况下 (如使用 10%Dodonew/10%000Webhost 和 30%000Webhost 作为训练集), 效果通常略优于其他方法, 这与文献 [26] 的结论一致. 随着训练集大小的增大, 不同方法的猜测成功率有不同程度的提高[1]. 其中,

[1]我们使用 FLA 源码, 分别使用原始论文中推荐的 "大模型" (隐藏层单元为 1000) 和 "小模型" (隐藏层单元为 200) 参数



(a) 50%Dodonew→10 万 Dodonew (b) 30%Dodonew→10 万 Dodonew (c) 10%Dodonew→10 万 Dodonew

(d) 50%000Webhost→10 万 000Webhost (e) 30%000Webhost→10 万 000Webhost (f) 10%000Webhost→10 万 000Webhost

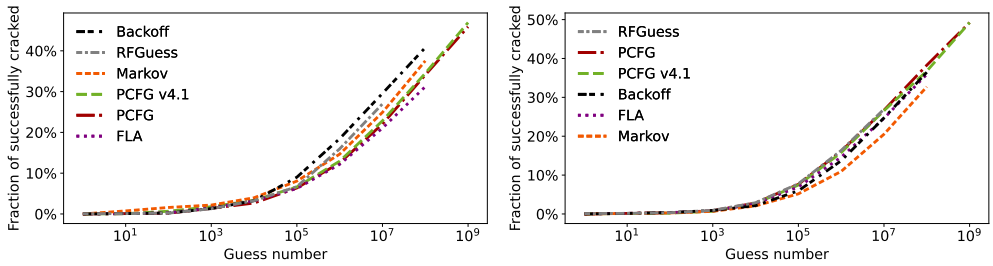
图 13 漫步猜测实验结果图 (训练集大小对不同方法的影响)

Figure 13 Experimental results of trawling guessing (Impact of training set size on different methods)

PCFG 的猜测成功率变化最大, 原因在于, PCFG 在生成猜测时, 结构段的填充内容 (如 S_3→!!!) 均来自训练集, 较小的训练集无法提供足够的填充片段.

此外, RFGuess [26] 在使用大规模数据集训练和生成口令时, 会占用大量的内存 (如使用 500 万训练集约占用 40–80 GB, 原因在于 RFGuess 在运行时会将所有字符前缀数据同时读入内存), 且生成速度较慢 (每秒可生成 130 个口令; 见表 10), 这使得该方法更适用于训练数据资源相对匮乏 (如和银行相关的口令数据) 和对生成速度要求不高的在线猜测场景.

口令策略对不同方法的影响如图 14 所示. 观察发现, 同一测试集, 口令策略不同, 攻破率也不同. 例如, Backoff [22] 在不经口令策略筛选的测试集上, 效果优于其他方法, 而在筛选之后, 猜测成功率与其他方法差异不大. 这表明, 在对口令猜测方法进行评估与对比时, 考虑训练集与测试集口令策略的一致性是很有必要的. 尽我们所知, 当前尚未有研究通过具体的实验探究过该问题.



(a) 80%000Webhost→100 万 Wishbone (b) 80%000Webhost→100 万 Wishbone((LD len≥6))

图 14 漫步猜测实验结果图 (口令策略对不同方法结果的影响)

Figure 14 Experimental results of trawling guessing (Impact of password policy on different methods)

4.4.2 基于个人信息的定向攻击场景

对于基于个人信息的定向猜测场景, 考虑到包含个人信息的数据集通常规模不大, 我们没有采用不同比例的划分方法, 而是遵循了之前在定向猜测研究中最常见的划分方式, 即将 50% 用于训练, 剩下的 50% 用于测试. Wang 等人 [27] 指出, 这种划分方式是合理的, 原因如下: 首先, 聪明的攻击者会不断改进他们的训练集, 使其分布尽可能与测试集接近. 例如, 攻击者可能会选择与攻击目标具有相同语言、服务类型和口令策略的数据集来进行训练. 而这种一半对一半的划分方式正好模拟了这种理想的攻击场景. 其次, 这种做法并不违反机器学习的原则, 即训练集和测试集应该不同. 根据口令服从 Zipf 分布的特性 [49,57],

重复了多次实验, 发现 “小模型” 的结果略优于 “大模型”, 一个可能的原因是, 50%Dodonew 作为训练集的数据量不足.

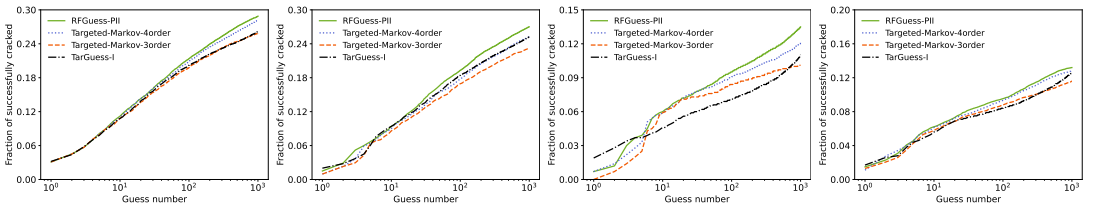
50% 用于训练的口令中会有相当一部分不会出现在剩余 50% 的测试口令中. 比如, 随机将 Dodonew 等分为两半 (记为训练集 Dodonew-tr 和测试集 Dodonew-ts), Dodonew-tr 中 81.22% 的口令没有出现在 Dodonew-ts 中. 这一法则确保了训练集和测试集相似但不相同. 事实上, 许多网站的口令泄露情况发生过多次, 如 Yahoo [83]、Flipboard [84]、Twitter [85] 和 Anthem [86] 等.

本文对三种最具代表性的基于个人信息的定向口令猜测算法进行了比较, 这些算法包括 TarGuess-I [28]、Targeted-Markov [60] 和 RFGuess-PII [26], 同时考虑了同站和跨站猜测场景 (实验设置详见表 11). 实验结果如图 15 所示. 总体而言, 这三种定向猜测算法的攻破率差距不大, 其中, RFGuess-PII 略微优于其他两种方法, 这与之前研究 [26] 的实验结果一致.

表 **11** 定向猜测攻击场景设置

Table 11 Experimental setup of targeted guessing scenarios

#	训练集	数目	测试集	数目	使用的个人信息种类
1	50% PII-12306	64 651	50% PII-12306	64 651	邮箱, 用户名, 姓名, 生日, 手机号
2			50% PII-Dodonew	80 758	
3	50% PII-ClixSense	1111 022	50% PII-ClixSense	1111 022	
4			50% PII-Rootkit	34 665	
5	Tianya→Dodonew (len≥8)	434 255	Tianya→CSDN (len≥8)	826,559	用户旧口令
6	LinkedIn→Yahoo (LD, len≥6)*	40 812	LinkedIn→000Webhost (LD, len≥6)	259 175	
7	80% COMB (邮箱前缀匹配)	342 921 727	20% COMB (邮箱前缀匹配)	85 730 432	



(a) PII-12306→PII-12306 (b) PII-12306→PII-Dodonew (c) PII-ClixSense→ClixSense (d) PII-ClixSense→PII-Rootkit

图 15 基于个人信息的定向口令猜测场景

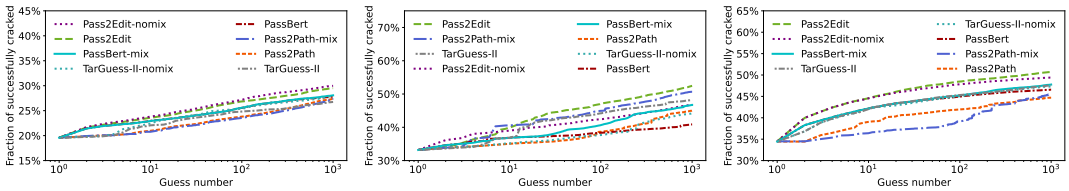
Figure 15 Targeted password guessing scenarios based on personal information

4.4.3 基于口令重用的定向攻击场景

对于口令重用攻击, 本文重现了文献 [35] 中的三个场景 (即对应表 11 中的场景 #5–#7), 对比了当前最先进的几种口令重用算法, 即 TarGuess-II [28]、Pass2Path [31]、PassBERT [32] 以及 Pass2Edit [35]. 其中, TarGuess-II 和 Pass2Edit 还使用了额外的流行口令字典 (预先使用 3 个数据集进行混合, 取频率相乘后排名靠前的 Top-10000 口令; 详见文献 [35] 的 4.3 节), 即在生成的重用口令中混入流行口令, 来模拟用户在创建新口令时选择流行口令的脆弱行为. 为了更公平地对比, 对 Pass2Path 和 PassBERT 也采用同样的流行字典进行混合, 混合方式为模型生成的重用口令: 流行口令 = 2 : 1 (该比例为实验测试的最佳比例). 同时, 为了排除混合方法不同带来的偏差, 还去除了 TarGuess-II 和 Pass2Edit 所使用的流行字典, 即 Pass2Edit-nomix 和 TarGuess-II-nomix.

根据图 16 和文献 [35] 的结果, 在 1000 猜测数下, Pass2Edit(或 Pass2Edit-nomix) [35] 的猜测成功率高于其他方法. 在英文数据集上, PassBERT [32] 和 PassBERT-mix [32] 在小猜测数 (<10) 下表现出了更好的猜测效果, 明显优于 TarGuess-II [28] 和 Pass2Path [31]. 随着猜测次数的增加, PassBERT 和 PassBERT-mix 的攻破率逐渐与 TarGuess-II 趋于持平. 而在中文数据集上, Pass2Path 在引入流行口令混合后 (即 Pass2Path-mix), 其猜测效果明显超过了 TarGuess-II 和 PassBERT. 在混合数据集上, 除了 Pass2Edit 之外, 其他方法的攻破率相对均衡, 没有明显的差异.

此外, 本文还统计了不同方法的训练时间、模型大小和口令生成速度, 如表 12 所示, 发现 TarGuess-II [28] 的训练速度最快, PassBERT [32] 的生成速度最快 (需要 GPU 支持), Pass2Edit [35] 的模型体积最小. 因此, 在实际应用时, 可优先尝试 Pass2Edit 及其变种, 但对于特定的数据集和猜测需求, 可以考虑其



(a) LinkedIn → 000Webhost (b) Tianya → CSDN (c) 80%COMB → 20%COMB

图 16 口令重用攻击场景

Figure 16 Targeted password guessing scenarios based on password reuse

他方法以获取最佳性能. 例如, 口令泄露检测服务 (支持口令变体泄露查询, 如文献 [87]) 每天通常会处理数百万用户的查询请求, 需要生成数以亿计的口令变体, 因而可能对算法的生成速度要求更高.

表 12 各个口令重用模型的计算效率 †

Table 12 Performance of different reuse models

模型	训练时间 (s)	模型大小	口令生成速度 (PW/s)
TarGuess-II [28]	0.56	1.0 G	1250
Pass2Path [31]	31	55.2 M	1470
Pass2Edit [35]	277	10.3 M	1538
PassBERT [32]	260	81.2 M	7143

†CPU: i7-9700 @ 3.00 GHz; GPU: GeForce RTX 2060 (使用 LinkedIn→000Webhost).

5 口令猜测算法应用

近年来, 涌现了大量口令猜测算法 [16,24,26,32,35], 这些算法的应用主要集中在防御者的视角, 特别是更准确地建模口令的抗猜测性 (guessability). 本节将深入探讨口令猜测算法在口令强度评估、数字取证以及口令泄露检测等领域的实际应用, 同时也将探讨它们所面临的挑战和问题.

5.1 口令强度评估

了解最先进的攻击技术对于设计最佳的防御策略至关重要. 口令作为许多安全系统的第一道防线甚至是唯一一道防线, 其强度直接影响着系统的安全性. 因此, 对口令强度进行准确评估尤为重要, 而口令猜测算法的一个核心应用即是从攻击者的视角准确评估口令强度. 当前, 已有一系列基于猜测算法的口令强度评价器 (PSM), 如 FLA-PSM [14]、fuzzyPSM [88]、CNN-PSM [89]、chunk_level-PSM [33] 等; 此外, 还有基于定向猜测算法的 PSM, 如 PPSM [31]. 然而, 出乎意料的是, 截至目前, 基于猜测算法的 PSM 均未在实际系统中得到广泛应用. 一个主要原因在于, 口令的强度不仅受口令本身复杂性的影响, 还受攻击者所掌握的信息影响. 因此, 要准确评估口令的强度, 需要考虑多种不同情境, 使用单一的 PSM 无法实现. 例如, 基于漫步猜测算法的 PSM 可能无法准确评估包含个人信息的口令的强度; 攻击者如果已经掌握了用户的旧口令, 那么该用户创建的新口令的强度也会受到影响. 如何以科学合理的方式充分利用各种猜测算法来准确评估用户口令的强度, 仍然是一个需要深入研究的问题.

5.2 数字取证

当前, 许多电子设备 (如手机和个人电脑) 都使用口令作为安全保护手段, 由此带来的设备解锁问题已成为数字取证工作中最为严峻的挑战之一 [90]. 若无法及时获取犯罪嫌疑人的口令, 可能会延缓调查进程, 甚至可能导致更多犯罪行为的发生. 幸运的是, 现有的口令猜测算法为执法机构破解加密设备/材料提供了一种潜在的途径, 帮助加速数字取证工作的进展. 例如, 执法人员可以通过收集开源情报 (OSINT) 来获取与目标相关的个人信息, 再将这些个人信息与猜测算法相结合, 可以显著提高执法机构解锁设备的效率, 进而为数字取证工作提供重要线索.

5.3 口令泄露检测

口令泄露检测主要分为用户和服务器两个层面. 在用户层面, 用户可以使用口令检测服务 (例如 “Pwned Passwords” [91]) 来查询自己的口令是否已经泄露. 考虑到有 26%~33% 的用户在创建口令时

会对已有口令做出简单的修改 [3], 泄露检测服务还应具备检测用户相似口令是否曾经泄露的功能. 确保这一功能的关键是准确测量口令之间的相似性, 以便判断哪些口令可能是用户查询口令的变体 [87]. 虽然有一系列用于测量字符串相似度的指标 (例如基于编辑距离的指标和基于余弦相似度的指标 [56]), 但它们都无法准确地建模用户的口令重用和修改行为. 因此, 一种直观的方法是从攻击者的角度出发, 利用口令重用模型/算法来根据用户输入的查询口令生成相似的变体, 然后结合一定的字符串相似度指标, 为用户提供更全面准确的口令泄露查询服务.

在服务器层面, 口令猜测算法可用于生成诱饵口令, 来达到口令文件泄露检测的目的 [27,34,92]. 所谓诱饵口令, 是指服务器为每个用户的真实口令生成一系列 ($k-1$ 个) "假口令", 与用户的真实口令存储在一起, 达到以假乱真的目的. 如此一来, 一旦口令文件发生泄露, 即使攻击者能恢复出全部口令, 仍需要从每个用户的 k 个口令中分辨出哪个是真实口令. 为此, 攻击者需要进行在线登录尝试, 一旦系统检测到有一定数量的假口令在尝试登录, 即表明口令文件可能已经发生泄露. 整个流程的关键在于, 能否生成以假乱真的诱饵口令. Wang 等人 [34] 的研究表明, 理想诱饵口令的分布应与用户真实口令的分布一致, 而组合多种不同类型的口令猜测算法被作者证明是行之有效的方法之一.

尽管 Honeywords 技术已被提出多年, 且有一系列相关研究 [27,34,92,93]. 但截至目前, 尚未有网站实际部署了此类系统. 可能原因如下: (1) Honeywords 系统很容易受到拒绝服务攻击 (DOS). 具体而言, 如果诱饵口令与用户真实口令的分布几乎一致, 会出现大量的流行诱饵口令, 攻击者很容易利用这些特征对网站发起 DOS 攻击, 引发系统误报. (2) Honeywords 系统需要把用户正确口令索引相关的信息存储在一个单独的、固化的、极简设计的计算机系统上, 并假设该系统是绝对安全的. 这既不现实, 也需要网站增加额外的部署设备, 并对现有系统进行相应的调整.

6 总结与展望

当前, 虽然无口令方案崭露头角 (如 Passwordless [94]), 用户可能减少对口令的直接接触, 但口令在可预见的未来仍将是最主要的身份认证手段, 无可替代 [8]. 因此, 口令将继续作为保障数十亿网民和信息系统安全的第一道防线, 口令猜测算法的研究对建立可持续的口令安全生态至关重要. 本文系统性总结了当前国内外的主要口令猜测算法, 并指出了存在的不足和值得进一步研究的方向. 下面探讨口令猜测未来的研究需求.

6.1 掩码猜测研究

在 IEEE S&P '21 中, Pasquini 等人 [16] 提出了条件口令猜测 (conditional password guessing) 的概念 (即掩码攻击). 它是指攻击者通过各种攻击手段 (如肩窥、键盘敲击回声等侧信道攻击) 获取了用户口令的部分信息 (如口令长度和/或部分口令字符), 然后利用口令模板 (如 `**mm*91`) 来帮助生成更有效的猜测集. 在 USENIX SEC '23 中, Xu 等人 [32] 针对此场景提出了更有效的 PassBERT 模型. 然而, 他们都没对掩码场景的现实性和危害性进行系统的讨论. 例如, 攻击者能通过哪些方式获取用户口令的哪些部分信息? 口令部分信息泄露会造成多大的危害? 当前, 尚未有关于掩码猜测场景的系统研究.

6.2 多口令猜测研究

在 IEEE S&P'19 中, Pal 等人 [31] 指出了一个有趣的现象: 通过将同一用户的两个旧口令输入到口令重用模型 Pass2Path 中, 然后合并生成的猜测集, 可以有效提高模型对同一用户新口令的猜测成功率. 然而, 这种方法并没有充分考虑到旧口令之间的内在联系. 此外, 研究作者并未提供一种有效的猜测混合顺序, 即在给定猜测次数的情况下, 如何对多个口令模型生成的猜测进行排序. 另一方面, 当尝试使用同一用户的多个旧口令来辅助猜测其目标口令时, 可能存在一个潜在问题, 即引入与目标口令完全不相关的口令. 这种情况可能会对模型的训练造成干扰, 甚至可能导致通过多口令训练的模型效果不如仅使用单个旧口令训练的模型, 即综合多个信息反而导致猜测成功率下降. 截至目前, 尚未有关于如何利用用户多个旧口令来提高猜测攻击成功率的系统研究. 然而, 大规模的数据泄露事件使得攻击者很容易获取同一用户的多个旧口令 (如含有邮箱的混合数据集中利用邮箱进行匹配), 因此, 设计利用用户多个旧口令的新型猜测方法/模型具有重要的现实意义.

6.3 基于隐式个人信息的口令猜测研究

当前关于定向口令猜测的研究尚处于初步阶段, 主要集中在如何利用显式个人信息 (即能直接作为口令的组成部分) 的层面. 主流的定向猜测算法 (如文献 [28]) 均是启发式构造用户个人信息的各种变体, 将不同类型的个人信息匹配成标签的形式, 然后再进行相应的训练和猜测生成. 然而这种方式存在固有缺陷, 例如, 很难穷举出用户个人信息的所有变体. 此外, 这种标签匹配的方法, 无法有效利用用户的隐式个人信息. 例如, 对于 000Webhost 的用户, 他们大多是网站管理员, 其安全意识显著高于普通用户群体. 如何利用 "网站管理员" 这一隐式身份信息 (群体信息) 是一个值得研究的问题. 事实上, 已有研究 (如文献 [28]) 表明隐式个人信息 (如教育背景、性别等) 确实会影响用户创建口令的行为. 但如何量化和有效地建模这种影响, 并将其引入口令猜测算法, 仍需进一步探索.

6.4 基于大模型的口令猜测研究

随着深度学习技术的不断发展, 大型语言模型如 GPT-4 [95] 等在自然语言处理任务中展现出了强大的能力. 这些模型具备了对文本数据的高度理解和生成能力, 因此可以用于更智能化的口令猜测攻击和防御. 具体而言, 大型模型能够生成更加真实且符合用户行为习惯的猜测. 它们可以分析用户的语言模式、兴趣爱好, 并利用个人信息构建个体的社交网络关系 (例如, 可以识别亲子关系的亲密度高于普通朋友的关系), 同时还能生成群体画像 (例如, 了解儿童用户的典型特征 [96]). 通过对这些通用性大模型在口令数据上进行微调, 有望更有效地利用用户画像 (包括各种隐式个人信息) 和社交网络关系, 从而提高口令猜测的成功率. 因此, 将大模型与口令猜测研究相结合, 将为口令安全研究带来新的可能性.

6.5 口令猜测算法组合研究

2021 年, Wang 等人 [73] 和 Parish 等人 [97] 的研究都表明, 组合不同类型口令模型所生成的猜测集的破解率通常高于单一猜测集. 然而, 这种简单的 (平均/随机) 混合方式只能得到某个固定猜测数下的破解成功率, 无法充分发挥不同模型的优势; 此外, 不同口令模型对应不同的概率空间, 对混合后字典的重新排序也是一个挑战. 当前, 随着深度学习技术的发展, 不断有新的猜测方法被陆续提出, 如何有效地利用各种算法来提高口令猜测的效率成为一个迫切需要深入研究的问题. 2021 年, Murray 等人 [98] 为解决该问题提供了新的思路: 基于多臂老虎机模型来建模口令猜测问题. 具体而言, 他们将每个猜测集视为一个老虎机, 通过每次猜测来学习更多有关目标口令分布的信息. 借助这些信息, 他们能够选择使用与目标口令分布最匹配的猜测集进行猜测, 从而最大化猜测收益. 2022 年, Han 等人 [99] 分析了现有猜测方法的不同特点, 提出了一个参数化混合猜测框架. 该框架由模型剪枝和最优猜测数分配策略构成, 能够利用不同方法的优势来生成更高效的猜测集. 期待更多的关于组合口令猜测算法的研究.

6.6 口令策略研究

为了促使用户创建更安全的口令, 网站往往会设置复杂的口令策略, 如长度必须不少于 8 位, 且必须包含两种以上字符类型. 这些要求看似提高了用户使用弱口令的门槛, 但用户在生成口令时遵循省力原则, 通常会对已有口令或流行口令做出简单修改, 来对规则进行适配. 复杂的口令策略既增加了用户的负担, 又未必显著提高口令的安全性. 如 Rao 等人的研究表明 [100], 仅仅增加口令长度, 不能显著提升口令的安全性; Wang 等人的研究表明 [101], 增加口令字符组成的种类, 不能提升口令的安全性. 事实上, 一些看似专门精心设计的口令策略往往存在不合理之处. 例如, 知名密码与信息安全期刊 ACM TOPS 的投稿网站在用户注册时要求口令长度大于等于 8 且至少包括两个数字 [102], 那么即使口令非常强健, 像 `iycFsgfsgado2.Ouhspr` 这样的口令, 仍然无法通过验证. 其本质原因是, 网站无法准确衡量用户口令的安全性, 进而无法给出科学合理的口令生成建议. 因此, 基于口令猜测算法和口令强度评测算法的研究进展, 对现有口令安全策略的系统性研究十分必要.

总之, 口令猜测研究是一个充满潜力的领域, 涉及多个学科的交叉知识, 包括密码学、统计学、自然语言处理和机器学习等, 十分具有挑战性. 这一研究领域不仅具有理论深度, 还具有广泛的实际应用价值. 随着计算技术的不断发展和大规模口令数据的不断泄露, 口令猜测攻击和防御之间的竞争将继续演化, 为研究者提供了丰富的机遇和挑战. 相信口令猜测研究将吸引更多学者的关注和深入研究, 为建设可持续的口令安全生态提供重要的理论指导和方法支撑.

参考文献

[1] AXEL B, BOUDHAYAN C, LENNIE D, et al. Security on the IBM mainframe[EB/OL]. 2014-12-01. https://www.redbooks.ibm.com/redbooks/pdfs/sg247803.pdf

[2] KEITH M, SHAO B, STEINBART P J. The usability of passphrases for authentication: An empirical field study[J]. International Journal of Human-Computer Studies, 2007, 65(1): 17–28. [DOI: 10.1016/j.ijhcs.2006.08.005]

[3] WANG P, WANG D, HUANG X Y. Advances in password security[J]. Journal of Computer Research and Development, 2016, 53(10): 2173–2188. [DOI: 10.7544/issn1000-1239.2016.20160483]
王平, 汪定, 黄欣沂. 口令安全研究进展 [J]. 计算机研究与发展, 2016, 53(10): 2173–2188. [DOI: 10.7544/issn1000-1239.2016.20160483]

[4] WANG D, WANG N, WANG P, et al. Preserving privacy for free: Efficient and probably secure two-factor authentication scheme with user anonymity[J]. Information Sciences, 2015, 321: 162–178. [DOI: 10.1016/j.ins.2015.03.070]

[5] YANG Y J, LU H B, LIU J K, et al. Credential wrapping: From anonymous password authentication to anonymous biometric authentication[C]. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (ASIA CCS '16). ACM, 2016: 141–151. [DOI: 10.1145/2897845.2897854]

[6] YubiKey 5 series[EB/OL]. 2023. https://www.yubico.com/products/yubikey-5-overview/

[7] WANG D. Research on Key Issues in Password Security[D]. Beijing: Peking University, 2017.
汪定. 口令安全关键问题研究 [D]. 北京: 北京大学, 2017.

[8] WANG D, CHEN X F, MA J F. The end of passwords[J]. Science Foundation in China, 2022, 36(3): 432–433+445. [DOI: 10.16262/j.cnki.1000-8217.2022.03.029]
汪定, 陈晓峰, 马建峰. 终结口令 [J]. 中国科学基金, 2022, 36(3): 432–433+445. [DOI: 10.16262/j.cnki.1000-8217.2022.03.029]

[9] BONNEAU J, HERLEY C, VAN OORSCHOT P C, et al. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes[C]. In: Proceedings of 2012 IEEE Symposium on Security and Privacy (SP '12). IEEE, 2012: 553–567. [DOI: 10.1109/SP.2012.44]

[10] ZIMMERMANN V, GERBER N. The password is dead, long live the password—A laboratory study on user perceptions of authentication schemes[J]. International Journal of Human-Computer Studies, 2020, 133: 26–44. [DOI: 10.1016/j.ijhcs.2019.08.006]

[11] EITEL B. Will passwords become a thing of the past?[EB/OL]. 2023-03-22. https://www.idsalliance.org/will-passwords-become-a-thing-of-the-past/

[12] BONNEAU J, HERLEY C, VAN OORSCHOT P C, et al. Passwords and the evolution of imperfect authentication[J]. Communications of the ACM, 2015, 58(7): 78–87. [DOI: 10.1145/2699390]

[13] MEYER B. COMB: Largest breach of all time leaked online with 3.2 billion records[EB/OL]. 2022-07. https://cybernews.com/news/largest-compilation-of-emails-and-passwords-leaked-free/

[14] MELICHER W, UR B, SEGRETI S M, et al. Fast, lean, and accurate: Modeling password guessability using neural networks[C]. In: Proceedings of the 25th USENIX Conference on Security Symposium (SEC '16). USENIX, 2016: 175–191.

[15] HITAJ B, GASTI P, ATENIESE G, et al. PassGAN: A deep learning approach for password guessing[C]. In: Applied Cryptography and Network Security—ACNS 2019. Springer Cham, 2019: 217–237. [DOI: 10.1007/978-3-030-21568-2_11]

[16] PASQUINI D, GANGWAL A, ATENIESE G, et al. Improving password guessing via representation learning[C]. In: 2021 IEEE Symposium on Security and Privacy (SP '21). IEEE, 2021: 1382–1399. [DOI: 10.1109/SP40001.2021.00016]

[17] MORRIS R, THOMPSON K. Password security: A case history[J]. Communications of the ACM, 1979, 22(11): 594–597. [DOI: 10.1145/359168.359172]

[18] John the Ripper password cracker[EB/OL]. 2023-10-01. https://www.openwall.com/john/

[19] Hashcat advanced password recovery[EB/OL]. 2023-10-01. https://hashcat.net/hashcat/

[20] NARAYANAN A, SHMATIKOV V. Fast dictionary attacks on passwords using time-space tradeoff[C]. In: Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS '05). ACM, 2005: 364–372. [DOI: 10.1145/1102120.1102168]

[21] WEIR M, AGGARWAL S, DE MEDEIROS B, et al. Password cracking using probabilistic context-free grammars[C]. In: Proceedings of 2009 30th IEEE Symposium on Security and Privacy(SP '09). IEEE, 2009: 391–405. [DOI: 10.1109/SP.2009.8]

[22] MA J, YANG W N, LUO M, et al. A study of probabilistic password models[C]. In: Proceedings of 2014 IEEE Symposium on Security and Privacy(SP '14). IEEE, 2014: 689–704. [DOI: 10.1109/SP.2014.50]

[23] VERAS R, COLLINS C, THORPE J. On semantic patterns of passwords and their security impact[C]. In: Proceedings of Network and Distributed System Security (NDSS) Symposium 2014. NDSS, 2014.
[24] PASQUINI D, CIANFRIGLIA M, ATENIESE G, et al. Reducing bias in modeling real-world password strength via deep learning and dynamic dictionaries[C]. In: Proceedings of 30th USENIX Security Symposium (SEC '21). USENIX, 2021: 821–838.
[25] DELL'AMICO M, MICHIARDI P, ROUDIER Y. Password strength: An empirical analysis[C]. In: Proceedings of the 29th Annual IEEE International Conference on Computer Communications (IEEE INFOCOM 2010). IEEE, 2010: 1–9. [DOI: 10.1109/INFCOM.2010.5461951]
[26] WANG D, ZOU Y K, ZHANG Z J, et al. Password guessing using random forest[C]. In: Proceedings of the 32nd USENIX Conference on Security Symposium (SEC '23). USENIX, 2023: 965–982.
[27] WANG D, CHENG H B, WANG P, et al. A security analysis of honeywords[C]. In: Proceedings of Network and Distributed System Security (NDSS) Symposium 2018. NDSS, 2018: 1-15.
[28] WANG D, ZHANG Z J, WANG P, et al. Targeted online password guessing: An underestimated threat[C]. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16). ACM, 2016: 1242–1254. [DOI: 10.1145/2976749.2978339]
[29] VERAS R, COLLINS C, THORPE J. A large-scale analysis of the semantic password model and linguistic patterns in passwords[J]. ACM Transactions on Privacy and Security (TOPS), 2021, 24(3): 1–21. [DOI: 10.1145/3448608]
[30] DELL'AMICO M, FILIPPONE M. Monte Carlo strength evaluation: Fast and reliable password checking[C]. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15). ACM, 2015: 158–169. [DOI: 10.1145/2810103.2813631]
[31] PAL B, DANIEL T, CHATTERJEE R, et al. Beyond credential stuffing: Password similarity models using neural networks[C]. In: Proceedings of 2019 IEEE Symposium on Security and Privacy (SP '19). IEEE, 2019: 417–434. [DOI: 10.1109/SP.2019.00056]
[32] XU M, YU J T, ZHANG X Y, et al. Improving real-world password guessing attacks via bi-directional Transformers[C]. In: Proceedings of the 32nd USENIX Conference on Security Symposium (SEC '23). USENIX, 2023: 1001–1018.
[33] XU M, WANG C, YU J, et al. Chunk-level password guessing: Towards modeling refined password composition representations[C]. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21). ACM, 2021: 5–20. [DOI: 10.1145/3460120.3484743]
[34] WANG D, ZOU Y K, DONG Q Y, et al. How to attack and generate honeywords[C]. In: Proceedings of 2022 IEEE Symposium on Security and Privacy (SP '22). IEEE, 2022: 489–506. [DOI: 10.1109/SP46214.2022.9833598]
[35] WANG D, ZOU Y K, XIAO Y-A, et al. PASS2EDIT: A multi-step generative model for guessing edited passwords[C]. In: Proceedings of the 32nd USENIX Conference on Security Symposium (SEC '23). USENIX, 2023: 983–1000.
[36] YU W, YIN Q S, YIN H, et al. A systematic review on password guessing tasks[J]. Entropy, 2023, 25(9): 1303. [DOI: 10.3390/e25091303]
[37] VIJAYAN J. RockYou hack exposes names, passwords of 30M accounts[EB/OL]. 2009-12-15. https://bitly.ws/Vo9A
[38] 2011 Chinese website user information leakage incident [EB/OL]. 2023-09-18.
2011 年中国网站用户信息泄露事件 [EB/OL]. 2023-09-18. https://bitly.ws/VnNK
[39] Dropbox hack 'affected 68 million users'[EB/OL]. 2014-12-25. https://www.bbc.com/news/technology-37232635
[40] Response to incident that 12306 was exposed to leak user information: Might be caused by ticket grabber plug-in[EB/OL]. 2016-08-31.
12306 被曝泄漏用户信息回应: 或是抢票插件所致 [EB/OL]. 2016-08-31. https://bitly.ws/Vo9I
[41] TAYLOR C. Twitch resets all user passwords after suffering data breach[EB/OL]. 2015-03-24. https://techcrunch.com/2015/03/23/twitch-passwords-data-breach-hack/
[42] SELYUKH C. Every Yahoo account that existed in mid-2013 was likely hacked[EB/OL]. 2017-10-03. https://bitly.ws/Vo9M
[43] Over 92 million MyHeritage emails containing hashed passwords breached[EB/OL]. 2018-06-06. https://bitly.ws/Vo9T
[44] Police are investigating the suspected leak of 500 million user information in Huazhu[EB/OL]. 2018-08-29.
华住 5 亿条用户信息疑遭泄露警方介入调查 [EB/OL]. 2018-08-29. https://bitly.ws/Vo9X
[45] MALIK P. Reddit-acquired dubsmash officially shut down[EB/OL]. 2019-02-18. https://bitly.ws/Vo9Z
[46] DINDER D. Zoom gets stuffed: Here's how hackers got hold of 500,000 passwords[EB/OL]. 2020-04-28. https://bitly.ws/Voa3

[47] GATLAN S. Hacker leaks full database of 77 million Nitro PDF user records[EB/OL]. 2021-01-20. https://bitly.ws/Voa6

[48] TOULAS B. Neopets says hackers had access to its systems for 18 months[EB/OL]. 2022-09-01. https://bitly.ws/Voa8

[49] WANG D, CHENG H B, WANG P, et al. Zipf's law in passwords[J]. IEEE Transactions on Information Forensics and Security, 2017, 12(11): 2776–2791. [DOI: 10.1109/TIFS.2017.2721359]

[50] ZHANG M L, ZHANG Q H, LIU W F. A method of password attack based on structure partition and string reorganization[J]. Chinese Journal of Computers, 2019, 42(4): 913–928. [DOI: 10.11897/SP.J.1016.2019.00913]
章梦礼, 张启慧, 刘文芬, 等. 一种基于结构划分及字符串重组的口令攻击方法 [J]. 计算机学报, 2019, 42(4): 913–928. [DOI: 10.11897/SP.J.1016.2019.00913]

[51] HAN W L, LI Z G, YUAN L, et al. Regional patterns and vulnerability analysis of Chinese web passwords[J]. IEEE Transactions on Information Forensics and Security, 2016, 11(2): 258–272. [DOI: 10.1109/TIFS.2015. 2490620]

[52] WANG D, WANG P, HE D B, et al. Birthday, name and bifacial-security: Understanding passwords of Chinese web users[C]. In: Proceedings of the 28th USENIX Conference on Security Symposium (SEC '19). USENIX, 2019: 1537–1555.

[53] LI Z G, HAN W L, XU W Y. A large-scale empirical analysis of Chinese web passwords[C]. In: Proceedings of the 23rd USENIX Conference on Security Symposium (SEC '14). USENIX, 2014: 559–574.

[54] LI Y, WANG H N, SUN K. A study of personal information in human-chosen passwords and its security implications[C]. In: Proceedings of the 35th Annual IEEE International Conference on Computer Communications (IEEE INFOCOM 2016). IEEE, 2016: 1–9. [DOI: 10.1109/INFOCOM.2016.7524583]

[55] XIA Z Y, YI P, LIU Y Y, et al. GENPass: A multi-source deep learning model for password guessing[J]. IEEE Transactions on Multimedia, 2019, 22(5): 1323–1332. [DOI: 10.1109/TMM.2019.2940877]

[56] DAS A, BONNEAU J, CAESAR M, BORISOV N, et al. The tangled web of password reuse[C]. In: Proceedings of Network and Distributed System Security (NDSS) Symposium 2014. NDSS, 2014: 1–15.

[57] HOU Z D, WANG D. New observations on Zipf's law in passwords[J]. IEEE Transactions on Information Forensics and Security, 2023, 18: 517–532. [DOI: 10.1109/TIFS.2022.3176185]

[58] ROWE A. Study reveals average person has 100 passwords[EB/OL]. 2023-03-21. https://tech.co/password-managers/how-many-passwords-average-person

[59] HOUSHMAND S, AGGARWAL S, FLOOD R. Next gen PCFG password cracking[J]. IEEE Transactions on Information Forensics and Security, 2015, 10(8): 1776–1791. [DOI: 10.1109/TIFS.2015.2428671]

[60] WANG D, WANG P. The emperor's new password creation policies: An evaluation of leading web services and the effect of role in resisting against online guessing[C]. In: Computer Security—ESORICS 2015, Part II. Springer Cham, 2015: 456–477. [DOI: 10.1007/978-3-319-24177-7_23]

[61] HRANICKÝ R, ZOBAL L, RYSAVY O, et al. Distributed PCFG password cracking[C]. In: Computer Security—ESORICS 2020, Part I. Springer Cham, 2020: 701–719. [DOI: 10.1007/978-3-030-58951-6_34]

[62] HAN W L, XU M, ZHANG J J, et al. TransPCFG: Transferring the grammars from short passwords to guess long passwords effectively[J]. IEEE Transactions on Information Forensics and Security, 2021, 16: 451–465. [DOI: 10.1109/TIFS.2020.3003696]

[63] LI S B, WANG Z Y, ZHANG R Y, et al. Mangling rules generation with density-based clustering for password guessing[J]. IEEE Transactions on Dependable and Secure Computing, 2022. [DOI: 10.1109/TDSC.2022.3217002]

[64] KLEIN D V. Foiling the cracker: A survey of, and improvements to Unix password security[C]. In: Proceedings of the USENIX Security Workshop, Summer 1990. USENIX, 1990: 5–14.

[65] THOMAS W. A real-world analysis of Kerberos password security[C]. In: Proceedings of Network and Distributed System Security (NDSS) Symposium 1999. NDSS, 1999: 1–10.

[66] DÜRMUTH M, ANGELSTORF F, CASTELLUCCIA C, et al. OMEN: Faster password guessing using an ordered Markov enumerator[C]. In: Engineering Secure Software and Systems—ESSoS 2015. Springer Cham, 2015: 119–132. [DOI: 10.1007/978-3-319-15618-7_10]

[67] UR B, SEGRETI S M, BAUER L, et al. Measuring real-world accuracies and biases in modeling password guessability[C]. In: Proceedings of the 24th USENIX Conference on Security Symposium (SEC '15). USENIX, 2015: 463–481.

[68] GOODIN D. Anatomy of a hack: How crackers ransack passwords like "qeadzcwrsfxv1331"[EB/OL]. 2013-05-28. https://bitly.ws/VwUH

[69] LIU E Z, NAKSNISHI A, GOLLA M, et al. Reasoning analytically about password-cracking software[C]. In: Proceedings of 2019 IEEE Symposium on Security and Privacy (SP '19). IEEE, 2019: 380–397. [DOI: 10.1109/SP.2019.00070]

[70] ZHANG H D, WANG C W, RUAN W Q, et al. Digit semantics based optimization for practical password cracking tools[C]. In: Proceedings of the 37th Annual Computer Security Applications Conference (ACSAC '21). ACM, 2021: 513–527. [DOI: 10.1145/3485832.3488025]

[71] DI CAMPI A M, FOCARD R, LUCCIO F L. The revenge of password crackers: Automated training of password cracking tools[C]. In: Computer Security—ESORICS 2022, Part II. Springer Chan, 2022: 317–336. [DOI: 10.1007/978-3-031-17146-8_16]

[72] XIE Z J, ZHANG M, YIN A Q, et al. A new targeted password guessing model[C]. In: Information Security and Privacy—ACISP 2020. Springer Cham, 2020: 350–368. [DOI: 10.1007/978-3-030-55304-3_18]

[73] WANG D, ZOU Y K, TAO Y, et al. Password guessing based on recurrent neural networks and generative adversarial networks[J]. Chinese Journal of Computers, 2021, 44(8): 1519–1534. [DOI: 10.11897/SP.J.1016.2021.01519]
汪定, 邹云开, 陶义, 等. 基于循环神经网络和生成式对抗网络的口令猜测模型研究 [J]. 计算机学报, 2021, 44(8): 1519–1534. [DOI: 10.11897/SP.J.1016.2021.01519]

[74] GOODFELLOW I, POUGET-ABADIE J, MIRZA M, et al. Generative adversarial nets[C]. In: Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS '14), Volume 2. Curran Associates, Inc., 2014: 2672–2680.

[75] TOLSTIKHIN I, BOUSQUET O, GELLY S, et al. Wasserstein auto-encoders[C]. In: Proceedings of 6th International Conference on Learning Representations (ICLR '18). 2018: 1–16.

[76] YANG K Y, HU X X, ZHANG Q H, et al. VAEPass: A lightweight passwords guessing model based on variational auto-encoder[J]. Computers & Security, 2022, 114: 102587. [DOI: 10.1016/j.cose.2021.102587]

[77] PAGNOTTA G, HITAJ D, DE GASPARI F, et al. Passflow: Guessing passwords with generative flows[C]. In: Proceedings of 2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, 2022: 251–262. [DOI: 10.1109/DSN53405.2022.00035]

[78] LU B, ZHANG X K, LING Z M, et al. A measurement study of authentication rate-limiting mechanisms of modern websites[C]. In: Proceedings of the 34th Annual Computer Security Applications Conference (ACSAC '18). ACM, 2018: 89–100. [DOI: 10.1145/3274694.3274714]

[79] GRASSI P A, FENTON J L, NEWTON E M, et al. NIST Special Publication 800-63B. Digital identity guidelines: Authentication and lifecycle management[R]. June 2017. [DOI: 10.6028/NIST.SP.800-63b]

[80] FLORÊNCIO D, HERLEY C, VAN OORSCHOT P C. An administrator's guide to Internet password research[C]. In: Proceedings of the 28th USENIX Conference on Large Installation System Administration (LISA'14). USENIX, 2014: 44–61.

[81] WANG D, SHAN X, DONG Q Y, et al. No single silver bullet: Measuring the accuracy of password strength meters[C]. In: Proceedings of 32nd USENIX Security Symposium (SEC '23). USENIX, 2023: 947–964.

[82] LIU P Y, BLOCKI J, BAI W J. Confident Monte Carlo: Rigorous analysis of guessing curves for probabilistic password models[C]. In: Proceedings of 2023 IEEE Symposium on Security and Privacy (SP '23). IEEE, 2023: 626–644. [DOI: 10.1109/SP46215.2023.10179365]

[83] PAI V. Yahoo discloses its second data breach in 4 months[EB/OL]. 2016-12-15. https://www.medianama.com/2016/12/223-yahoo-2nd-data-breach/

[84] Flipboard confirms it was hacked twice: 150M users at risk as passwords stolen[EB/OL]. 2019-05-29. https://bit.ly/3ICqJ0S

[85] HEILIGENSTEIN M X. Twitter data breaches: Full timeline through 2023[EB/OL]. 2023-10-5. https://firewalltimes.com/twitter-data-breach-timeline/

[86] EIDE N. Anthem breached again after contractor emailed file with 18,500 members' info[EB/OL]. 2017-08. https://www.ciodive.com/news/anthem-breached-again-after-contractor-emailed-file-with-18500-members-in/448334/

[87] PAL B, ISLAM M, BOHUK M S, et al. Might I get pwned: A second generation compromised credential checking service[C]. In: Proceedings of 31st USENIX Security Symposium (SEC '22). USENIX, 2022: 1831-1848.

[88] WANG D, HE D B, CHENG H B, et al. fuzzyPSM: A new password strength meter using fuzzy probabilistic context-free grammars[C]. In: Proceedings of 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '16). IEEE, 2016: 595–606. [DOI: 10.1109/DSN.2016.60]

[89] PASQUINI D, ATENIESE G, BERNASCHI M. Interpretable probabilistic password strength meters via deep learning[C]. In: Computer Security—ESORICS 2020, Part I. Springer Cham, 2020: 502–522. [DOI: 10.1007/978-3-030-58951-6_25]

[90] KANTA A, COISEL I, SCANLON M. A survey exploring open source Intelligence for smarter password cracking[J]. Forensic Science International: Digital Investigation, 2020, 35: 301075. [DOI: 10.1016/j.fsidi.2020.301075]

[91] Pwned passwords[EB/OL]. 2023-10-01. https://haveibeenpwned.com/Passwords

[92] JUELS A, RIVEST R L. Honeywords: Making password-cracking detectable[C]. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (CCS '13). ACM, 2013: 145–160. [DOI: 10.1145/2508859.2516671]

[93] ERGULER I. Achieving flatness: Selecting the honeywords from existing user passwords[J]. IEEE Transactions on Dependable and Secure Computing, 2015, 13(2): 284–295. [DOI: 10.1109/TDSC.2015.2406707]

[94] BRAND C, KARRA S. The beginning of the end of the password[EB/OL]. 2023-05-03. https://blog.google/technology/safety-security/the-beginning-of-the-end-of-the-password/

[95] OPEN AI. GPT-4 technical report[EB/OL]. 2023-10-01. https://cdn.openai.com/papers/gpt-4.pdf

[96] THEOFANOS M, CHOONG Y Y, MURPHY O. 'Passwords keep me safe'—Understanding what children think about passwords[C]. In: Proceedings of 30th USENIX Security Symposium (SEC '21). USENIX, 2021: 19–35.

[97] PARISH Z, CUSHING C, Aggarwal S, et al. Password guessers under a microscope: An in-depth analysis to inform deployments[J]. International Journal of Information Security, 2022, 21(2): 409–425. [DOI: 10.1007/s10207-021-00560-9]

[98] MURRAY H, MALONE D. Choosing wordlists for password guessing: An adaptive multi-armed bandit approach[C]. In: Foundations and Practice of Security—FPC 2021. Springer Cham, 2021: 393–413. [DOI: 10.1007/978-3-031-08147-7_27]

[99] HAN W L, ZHANG J J, XU M, et al. Parameterized hybrid password guessing method[J]. Journal of Computer Research and Development, 2022, 59(12): 2708–2722. [DOI: 10.7544/issn1000-1239.20210456]
韩伟力, 张俊杰, 徐铭, 等. 参数化混合口令猜测方法 [J]. 计算机研究与发展, 2022, 59(12): 2708–2722. [DOI: 10.7544/issn1000-1239.20210456]

[100] RAO A, JHA B, KINI G. Effect of grammar on security of long passwords[C]. In: Proceedings of the Third ACM Conference on Data and Application Security and Privacy (CODASPY '13). ACM, 2013: 317–324. [DOI: 10.1145/2435349.2435395]

[101] WANG C W, ZHANG J J, XU M, et al. # segments: A dominant factor of password security to resist against data-driven guessing[J]. Computers & Security, 2022, 121: 102848.

[102] Submission website of ACM Transactions on Privacy and Security[EB/OL]. 2024-02-01. https://mc.manuscriptcentral.com/tops

作者信息

邹云开 (1995–), 博士研究生. 主要研究领域为口令安全、机器学习.
zouyunkai@mail.nankai.edu.cn

汪定 (1985–), 教授, 全国密码专业学位教育指导委员会委员. 主要研究领域为密钥安全.
wangding@nankai.edu.cn