Computer Science and Application 计算机科学与应用, 2025, 15(3), 170-179
Published Online March 2025 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2025.153069

Hans汉斯

基于大数据的网络空间资产探测系统

陈明昊，徐亚峰，李　琳，王铭砾，姬懿轩

徐州工程学院信息工程学院(大数据学院)，江苏　徐州

收稿日期：2025年2月23日；录用日期：2025年3月20日；发布日期：2025年3月26日

摘　要

随着网络空间规模的不断扩展以及网络攻击技术的日益复杂化，如何高效地识别、整理和分析网络资产成为网络安全领域的重要研究方向。本文提出了一种基于大数据技术的网络空间资产探测系统，涵盖子域名扫描与活跃性测试、URL资产整理与去重、IP识别、指纹扫描及智能辅助分析等功能模块。系统采用分布式存储和高性能计算技术，以布隆过滤器优化数据去重流程，提升探测效率与精度。实验表明，该系统能够在大规模网络资产探测中实现快速、准确的分析，适用于动态复杂的网络环境。

关键词

网络空间资产探测，大数据技术，布隆过滤器，指纹识别，网络安全

Big Data-Based Cyberspace Asset Detection System

Minghao Chen, Yafeng Xu, Lin Li, Mingli Wang, Yixuan Ji

College of Information Engineering (Big Data College), Xuzhou University of Technology, Xuzhou Jiangsu

Received: Feb. 23rd, 2025; accepted: Mar. 20th, 2025; published: Mar. 26th, 2025

Abstract

As cyberspace continues to expand and cyberattacks become increasingly sophisticated, efficiently identifying, organizing, and analyzing cyber assets has become a critical research focus in the field of cybersecurity. This paper proposes and implements a big data-based cyberspace asset detection system that incorporates modules for subdomain scanning, URL asset processing, IP detection, fingerprint recognition, and intelligent analysis. Leveraging distributed storage and high-performance algorithms such as Bloom filters, the system improves detection accuracy and efficiency. Experimental results demonstrate the system's capability to perform rapid and accurate asset analysis in large-scale network environments, making it suitable for dynamic and complex scenarios.

文章引用: 陈明昊, 徐亚峰, 李琳, 王铭砾, 姬懿轩. 基于大数据的网络空间资产探测系统[J]. 计算机科学与应用, 2025, 15(3): 170-179. DOI: 10.12677/csa.2025.153069

Keywords

Cyberspace Asset Detection, Big Data Technology, Bloom Filter, Fingerprint Recognition, Cybersecurity

1. 引言

随着网络空间规模的不断扩大和网络攻击技术的日益复杂化，网络空间资产探测已成为网络安全领域的重要研究方向。传统的探测方法在处理海量数据时存在效率瓶颈，因此，基于大数据技术的网络空间资产探测系统应运而生。该系统通过分布式存储、高效算法和数据挖掘技术，能够高效识别、整理和分析子域名、IP 地址、URL 及系统指纹等网络资产，极大提高了数据处理能力和探测效率。通过结合分布式计算框架和智能化的数据分析手段，系统能够实时发现潜在的安全威胁，为应对复杂多变的网络安全挑战提供了创新性的技术支持，成为现代网络安全防护的重要组成部分[1] [2]。

2. 技术特点

2.1. Hadoop 分布式文件系统(HDFS)的架构与特点分析

Hadoop 分布式文件系统(HDFS)是为大数据存储和处理设计的分布式文件系统，具有高容错性、可扩展性和高吞吐量。它广泛应用于需要处理大量数据的场景，适合存储大文件并支持高效的并行读取和写入。下面从五个方面详细分析 HDFS 的特点和功能：

1) 架构设计

HDFS 采用主从架构，包含 NameNode 和 DataNode 两个核心组件。NameNode 负责管理元数据和文件系统结构，而 DataNode 负责实际的数据存储和块的管理。HDFS 通过这种架构支持高效的数据存储和管理，同时也能保证系统的扩展性和容错性。

2) 数据块与分布式存储

HDFS 将文件拆分成多个大小固定的数据块，并将这些块分布存储在集群中的多个 DataNode 上。每个数据块有多个副本，默认副本数为 3，保证数据的高可用性和容错性。通过分布式存储和副本机制，HDFS 能够有效地进行海量数据存储和管理。

3) 容错性与副本管理

HDFS 通过副本机制保证数据的容错性。即使某个 DataNode 发生故障，其他副本仍然可以提供数据，保证数据的可靠性。HDFS 支持 Rack Awareness 机制，使得副本分布跨多个机架，提高系统的容错能力，防止由于机架级故障导致数据丢失。

4) 数据读写模式

HDFS 采用写一次、读多次的模式，适合大规模数据集的存储和批处理操作。文件写入时被分割成多个数据块，按照顺序写入各个 DataNode，读操作则通过 NameNode 返回数据块的位置，客户端直接从 DataNode 读取数据。该设计保证了顺序数据的高效读取与写入，但不适合频繁的随机读写操作。

5) 扩展性与集群管理

HDFS 支持水平扩展，用户可以通过添加新的 DataNode 来增加存储容量和计算能力。集群管理和监

控通过 Hadoop 的相关工具进行，确保集群的健康与负载均衡。HDFS 自动进行数据块的重新分配和副本管理，确保集群在扩展或发生故障时能够高效运作。

通过这些特点，HDFS 能够为大规模数据处理提供可靠、高效的存储解决方案，适用于大数据分析、数据挖掘等多个领域。

2.2. 布隆过滤器特点分析

布隆过滤器(Bloom Filter)是一种高效的概率型数据结构，用于判断一个元素是否属于某个集合。它的特点主要体现在空间效率和查询速度上，但也存在一定的误判概率。以下是布隆过滤器的几个主要特点分析[3]：

1) 空间效率

布隆过滤器使用位数组和多个哈希函数，能够在空间上以非常高效的方式存储大量元素，而不需要实际存储所有的元素。相比传统的集合或哈希表，布隆过滤器能够显著减少内存的使用。

2) 查询时间常量

布隆过滤器的查询操作非常快速，查询时间是常数时间 $O(k)$，其中 k 是哈希函数的个数。这使得布隆过滤器非常适合用于高性能要求的应用场景，如大规模数据处理和实时查询。

3) 误判概率(假阳性)

布隆过滤器可能会误判一个元素存在于集合中，但绝不会错误地认为一个元素不在集合中。换句话说，如果布隆过滤器判断某个元素存在，实际上它可能并不存在于集合中，这种现象被称为“假阳性”。误判的概率与位数组的大小、哈希函数的个数以及已存储元素的数量有关。

4) 不可删除元素

一旦元素被添加到布隆过滤器中，它就无法从中删除。这是因为布隆过滤器是基于位数组和哈希函数的，删除操作可能会影响其他元素的正确性。如果需要删除元素，可以使用计数布隆过滤器，它在布隆过滤器的基础上增加了计数功能，允许元素的删除。

5) 可调节的误判率

通过合理选择位数组的大小和哈希函数的个数，可以控制误判率的大小。通常，通过增加位数组的长度和哈希函数的个数，可以降低误判率。然而，过多的哈希函数会导致位数组中的位设置过多，增加存储成本。

6) 并行处理和分布式支持

布隆过滤器本身支持并行查询，可以并行处理大量数据。此外，它也非常适合在分布式系统中使用，可以在多个节点上分布式存储和查询，从而处理海量数据。

2.3. 布隆过滤器算法

$$P = \left(1 - \mathrm{e}^{-kn/m}\right)^k \quad \text{(公式 1)}$$

n：插入到布隆过滤器中的元素数量。即您希望存储的元素的数量。

m：布隆过滤器的位数组长度。位数组越大，存储空间越多，可以容纳更多元素，从而影响误判率。

k：布隆过滤器使用的哈希函数的数量。每个元素通过多个哈希函数映射到位数组中的多个位置。

此公式计算了布隆过滤器的误判率，或称为假阳性率。假阳性率是指布隆过滤器错误地认为一个元素已经存在于集合中的概率。换句话说，布隆过滤器可能会返回一个“存在”的结果，尽管该元素其实并没有被插入。

公式含义：

当 k (哈希函数个数)越大时，误判率一般会增加，直到达到某个最大值。

公式中的 $e^{-kn/m}$ 表示随着元素数量 n 和位数组长度 m 增加，位数组中的“空闲空间”逐渐减少，误判率会增加。

为了降低误判率，通常需要调整 m 和 k。

最优哈希函数数量公式：

$$k^* = \frac{m}{n}\ln 2 \quad \text{(公式 2)}$$

此公式给出了一个理想的哈希函数数量 k^*，使得误判率最小。在此条件下，布隆过滤器在最少空间和哈希函数数量下实现了最优性能。

3. 系统设计与实现

3.1. 布隆过滤器算法设计

布隆过滤器的原理如图 1 所示，具体的实现步骤如下：① 初始化布隆过滤器：首先需要实例化一个布隆过滤器对象，指定其大小(Size)和哈希函数的数量(hash_count)。布隆过滤器内部会创建一个位数组，并将所有位初始化为 0。② 添加元素：在添加元素时，布隆过滤器会使用多个哈希函数(由 hash_count 确定)，对每个元素进行哈希处理。每个哈希函数的输出值会根据位数组的大小进行取模运算，以确定位数组中需要设置为 1 的位置。通过这种方式，布隆过滤器将元素映射到位数组中。③ 检查元素：当需要检查一个元素是否存在时，布隆过滤器会使用与添加元素时相同的哈希函数计算该元素的哈希值，并检查相应位置的位置是否为 1。如果所有哈希值对应的位置都为 1，则认为该元素可能已经存在于布隆过滤器中(可能是误判)。如果任意位置为 0，则说明该元素一定不在布隆过滤器中。④ 处理输入文件：在去重操作中，布隆过滤器会遍历输入文件中的每一行数据(如 URL)。对于每一行，布隆过滤器会检查该行是否已经存在。如果该行不在布隆过滤器中，则将其添加到过滤器并将其保存到输出文件中。⑤ 输出去重结果：最后，将所有未被布隆过滤器识别为重复的元素写入到输出文件中，完成去重过程[4]。

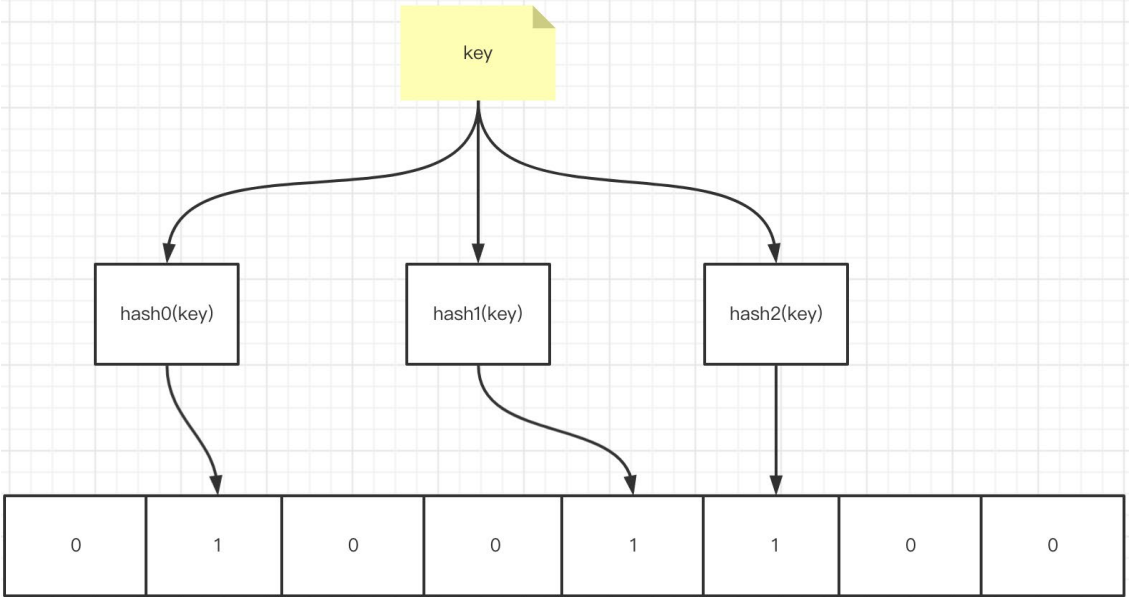

Figure 1. Example diagram of bloom filter principle
图 1. 布隆过滤器原理示例图

3.2. HDFS 的设计

分布式分布式储存如图 2 所示，具体步骤如下：

① 客户端请求上传文件

客户端通过 HDFS API 向 NameNode 请求上传文件。NameNode 是 HDFS 的主节点，负责元数据管理，包括文件的目录结构、文件块信息等。

② NameNode 分配块并返回 DataNode 信息

NameNode 根据文件大小将文件划分为多个数据块(默认块大小为 128 MB)，并为每个数据块分配存储 DataNode 的节点列表。

③ 数据流转移至 DataNode

客户端开始通过管道将文件分块发送到指定的 DataNode。每个数据块在传输时会被自动复制到其他 DataNode 上，以实现冗余存储(默认三副本策略)。

④ DataNode 确认接收数据块

当一个数据块被写入到所有副本的 DataNode 后，DataNode 会向客户端和 NameNode 发送成功确认信息。

⑤ 更新元数据

NameNode 在接收到所有数据块的确认信息后，更新文件元数据，包括文件路径、块列表、存储位置等，并返回给客户端上传成功的消息[5]。

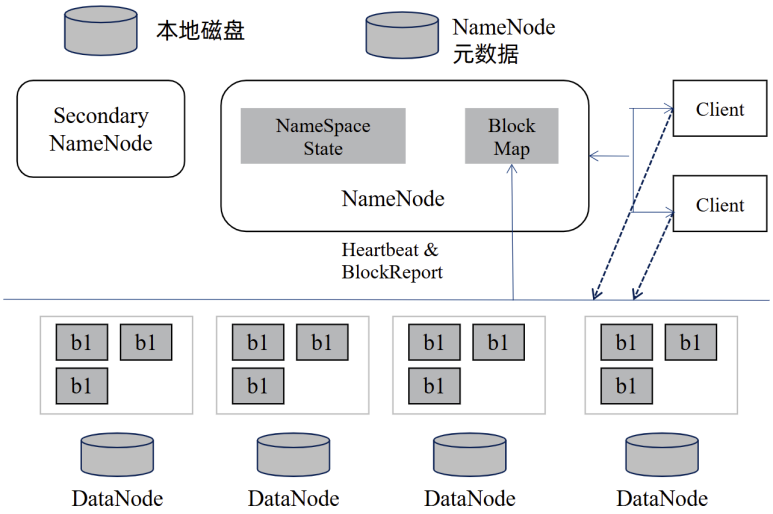

Figure 2. Distributed storage example diagram
图 2. 分布式储存示例图

3.3. 数据收集处理流程设计

1) URL 信息收集处理

本部分主要负责利用历史 URL 收集工具对目标域名进行 URL 资产的全面扫描与记录。通过调用 wayback URLs 和 Gau 工具，提取目标域名在公共存档和网络爬取中的历史 URL 数据。函数 run_wayback-urls (domain)和 run_gau (domain)分别封装了工具的执行流程，包含路径初始化、输出文件创建和命令执行等步骤。工具的运行采用 subprocess. Popen 模块调用，并将扫描结果存储至本地。程序内集成了对工具路径有效性及执行结果状态的校验机制，在路径配置错误或运行异常时，能够实时输出错误日志。通

过调用 geturl (domain)函数，用户可对指定域名执行 URL 扫描，生成详细的历史 URL 数据，为后续的漏洞扫描及攻击面分析提供详实的基础数据支持。

首先，通过 Wayback Archiver 和 Gau 工具收集历史和现存的 URL 数据，并存储到分布式文件系统(HDFS)中。随后，利用布隆过滤器对重复 URL 进行去重，确保处理效率与数据质量。接着，通过 filter_value 模块提取参数值，通过 filter_path 模块解析路径信息，将结果进一步规范化。最终，系统构建了参数字典和路径字典，为后续的安全检测任务提供了精确的输入源，URL 收集处理流程图如图 3 所示，该设计既提升了数据处理效率，又确保了提取信息的准确性与全面性，适用于大规模的 URL 分析与安全研究场景，系统实现如图 4 所示。

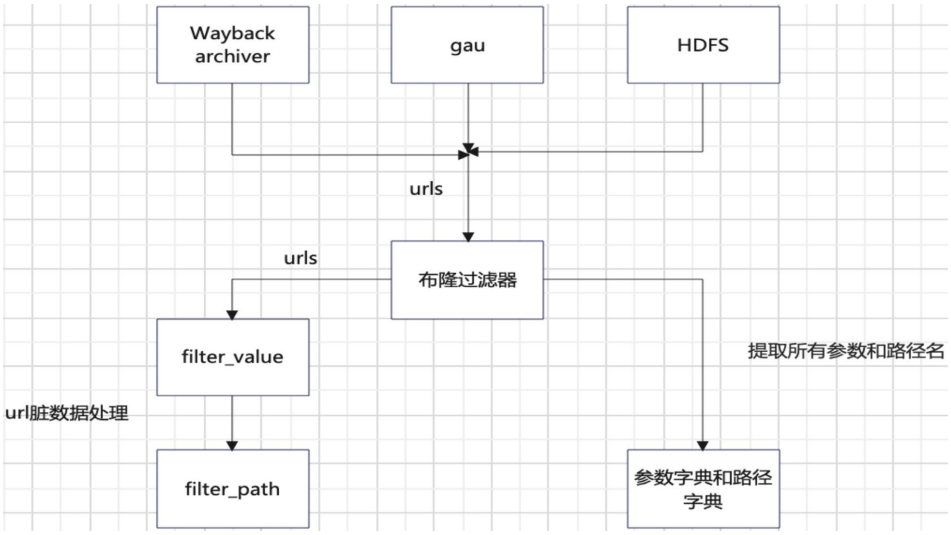

Figure 3. URL collection and processing flowchart
图 3. URL 收集处理流程图

Figure 4. URL scanning system implementation diagram
图 4. URL 扫描系统实现图

2) 子域名信息收集测活

子域名扫描模块利用开源工具 subfinder 对目标域名进行子域名枚举，旨在发现潜在的攻击面，增强对目标网络的全面侦察能力。通过封装 run_subfinder (domain)函数调用 subfinder 执行扫描并返回结果，同时通过 getsub (domain)从 HDFS 获取已存在的子域名数据，模块能够自动化执行子域名扫描与结果处理。获取到的子域名会经过布隆过滤器过滤，布隆过滤器通过高效地筛选出重复和无用子域名，减少冗余信息并加速后续处理。在此基础上，使用 httpx 模块对剩余子域名进行活跃度测试，httpx 能够并行化地发起多个 HTTP 请求，快速检测哪些子域名是可达的，从而精确识别出活跃的子域名，为后续的渗透测试和安全分析提供更有价值的数据。

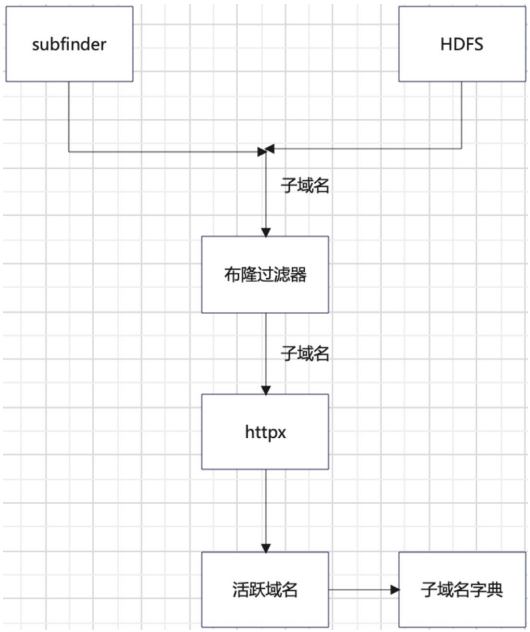

Figure 5. Subdomain collection and processing flowchart
图 5. 子域名收集处理流程图

Figure 6. Domain scanning system implementation diagram
图 6. 子域名扫描系统实现图

首先利用 subfinder 工具采集子域名数据并存储至分布式文件系统(HDFS)，随后通过布隆过滤器快速去重，提升数据处理效率。接着，采用 httpx 工具验证子域名的可达性，筛选出活跃域名，最终生成精准的子域名字典，供后续漏洞扫描等任务使用。该设计结合高效的去重机制与精准的活跃性验证，具备高性能和可扩展性，适用于大规模网络空间安全检测场景，子域名收集处理流程图如图 5 所示，实现界面如图 6 所示。

3) IP 扫描识别

该部分的设计目的是通过对目标域名解析出的 IP 地址进行扫描，识别开放端口及其服务信息。首先，利用 DNS 解析技术获取目标域名的相关 IP 地址，包括 A 记录、AAAA 记录及 CNAME 记录等类型，确保尽可能全面地收集关联的 IP 数据。随后，通过调用 nmap 工具对获取的 IP 地址列表逐一进行端口扫描，采用非特权扫描方式(-sT -sV)，以探测每个 IP 的开放端口及其运行的服务类型，同时记录端口的状态信息。扫描结果会根据每个 IP 进行归类和格式化处理，最终输出至日志文件，便于后续分析和存档。在实际执行过程中，为提高系统的健壮性，加入了对域名无解析记录、IP 扫描异常等情况的处理逻辑，从而保证整个扫描流程的顺利完成。

首先利用 DNS 解析技术获取目标域名的 IP 地址，包括 A 记录、AAAA 记录和 CNAME 记录，以确保全面收集关联的 IP 数据。接着，使用 python 的 nmap 库对这些 IP 地址逐一进行非特权端口扫描(-sT -sV)，探测开放端口及其服务信息，同时记录端口状态。扫描结果按照每个 IP 归类并格式化输出至日志文件，便于后续分析与存档。为了增强系统健壮性，设计中加入了对域名无解析记录、IP 扫描异常等情况的处理逻辑，确保整个扫描流程的稳定性和完整性，实现界面如图 7 所示。

Figure 7. IP scanning system implementation diagram
图 7. IP 扫描系统实现图

4) 指纹识别

通过调用 Nuclei 工具对目标域名进行技术指纹识别，利用 subprocess. Popen 模块执行 Nuclei 命令，指定目标域名为扫描对象，使用 http/technologies 模板提取目标的技术信息。扫描结果以 JSON 格式保存，便于后续分析与处理。程序中包含对工具路径的检查和异常处理，确保 Nuclei 工具路径正确，若出现路径错误或其他执行异常，会输出错误提示信息。通过调用 get fingerprint ()函数并传入域名，即可完成指纹信息的识别与保存。

通过调用 Nuclei 工具对目标域名进行技术指纹识别，利用 subprocess. Popen 执行命令并指定 http/technologies 模板进行扫描，结果以 JSON 格式保存，便于后续分析。在实现过程中，加入了对 Nuclei 工具路径的检查和异常处理，确保工具路径正确并能够顺利执行。通过封装 get fingerprint ()函数，传入域名即可完成技术指纹的识别与保存，实现界面如图 8 所示[6]。

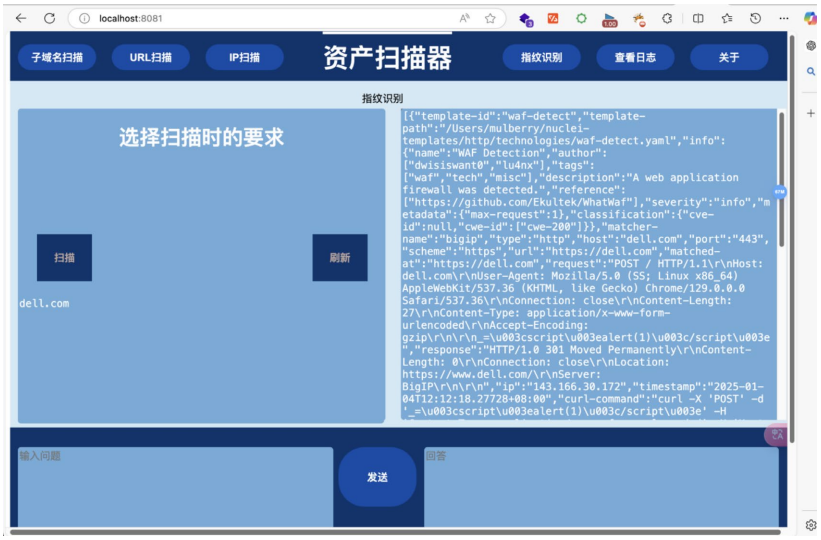

Figure 8. Fingerprint recognition system implementation diagram
图 8. 指纹识别系统实现图

3.4. 性能比较

为了展示本系统的优势，与第三方系统进行扫描结果的比对。首先测试 URL 扫描，针对同一个域名 dell.com，与常用的 URL 收集器 Wayback URLs 进行扫描结果的对比，本系统的扫描结果为 11,885 个，而 Wayback URLs 收集到的结果为 9730 个，根据结果可以看出，对于 URL 收集，本系统的结果要优于 Wayback URLs。然后测试子域名扫描，扫描目标同样为 dell.com，与站长工具中的子域名扫描引擎进行对比，本系统扫描结果为 2536 个子域名，站长工具中的工具扫描结果则为 59 个子域名，根据结果可以看出本系统在子域名扫描的能力要优于站长工具，具体对比结果如图 9 所示。

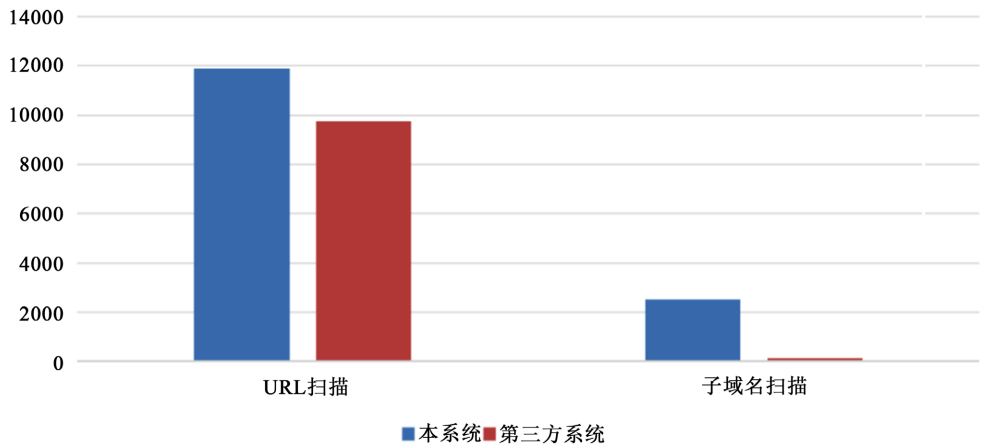

Figure 9. Performance comparison chart
图 9. 性能比较图

4. 结论

本文提出并设计了一个基于大数据的网络空间资产扫描器，旨在提高网络安全检测的效率和准确性。通过结合多个先进的技术手段，包括子域名扫描、URL 资产扫描、IP 识别、指纹识别及 GPT 指导功能等，本系统能够有效地识别和分析网络资产，发现潜在的安全风险。系统的核心优势在于其高效的数据存储与管理，利用 HDFS 的分布式架构保证了大规模数据的快速处理与存取，同时通过布隆过滤器和去重算法显著提高了数据的处理速度和准确性。

结果表明，系统能够快速扫描并识别出大量的网络资产，去重效果显著，能够有效降低误判率。GPT 指导功能为用户提供了实时的安全问题解答，进一步提升了系统的易用性和智能化水平。然而，系统仍存在一些局限性，如在特定复杂场景下的误判率和扫描时间的问题。未来的研究可以进一步优化算法，提高系统的稳定性和准确性，拓展更多扫描功能，支持更复杂的网络环境。本系统提供了一种高效、可靠且易于扩展的网络空间资产扫描解决方案，具有广泛的应用前景，对于大规模网络环境下的安全测试和资产管理具有重要意义。

基金项目

本文是徐州工程学院大学生创新创业项目(xcx2024193)的阶段性成果之一。

参考文献

[1] 澹台栋良. 面向网络空间安全的搜索引擎研究与实现[D]: [硕士学位论文]. 西安: 西安电子科技大学, 2021. https://doi.org/10.27389/d.cnki.gxadu.2020.000877

[2] 雷雪. Web 应用安全-漏洞扫描器的设计与实现[D]: [硕士学位论文]. 西安: 西安电子科技大学, 2021. https://doi.org/10.27389/d.cnki.gxadu.2020.002271

[3] 张鹏, 罗文华. 基于布隆过滤器查找树的日志数据区块链溯源机制[J]. 信息网络安全, 2024, 24(11): 1739-1748.

[4] 魏迎. 基于 Hadoop 技术的大规模数据分布式存储技术研究[J]. 自动化与仪器仪表, 2024(10): 182-186. https://doi.org/10.14016/j.cnki.1001-9227.2024.10.182

[5] 薛梅婷, 俞万刚, 张纪林, 曾艳, 袁俊峰, 周丽. 一种基于动态空间划分和压缩布隆过滤器相结合的分布式元数据负载均衡算法[J]. 计算机工程与科学, 2024, 46(8): 1381-1389.

[6] 吕宝路, 梁景普, 欧翰琪, 陈涛. 面向敏感信息检测的 Web 综合漏洞扫描器实现[J]. 电脑知识与技术, 2020, 16(23): 30-32. https://doi.org/10.14004/j.cnki.ckt.2020.2458