The Silent Spill: Measuring Sensitive Data Leaks Across Public URL Repositories

Tarek Ramadan
tarekramadan204@gmail.com
Concordia University
Montreal, QC, Canada

AbdelRahman Abdou
abdou@scs.carleton.ca
Carleton University
Ottawa, ON, Canada

Mohammad Mannan
m.mannan@concordia.ca
Concordia University
Montreal, QC, Canada

Amr Youssef
amr.youssef@concordia.ca
Concordia University
Montreal, QC, Canada

ABSTRACT

A large number of URLs are made public by various platforms for security analysis, archiving, and paste sharing—such as VirusTotal, URLScan.io, Hybrid Analysis, the Wayback Machine, and RedHunt. These services may unintentionally expose links containing sensitive information, as reported in some news articles and blog posts. However, no large-scale measurement has quantified the extent of such exposures. We present an automated system that detects and analyzes potential sensitive information leaked through publicly accessible URLs. The system combines lexical URL filtering, dynamic rendering, OCR-based extraction, and content classification to identify potential leaks. We apply it to 6,094,475 URLs collected from public scanning platforms, paste sites, and web archives, identifying 12,331 potential exposures across authentication, financial, personal, and document-related domains. These findings show that sensitive information remains exposed, underscoring the importance of automated detection to identify accidental leaks.

CCS CONCEPTS

• **Security and privacy** → *Web application security*.

KEYWORDS

data leakage, sensitive information exposure, URL analysis, web measurement, security automation, privacy risks, public datasets

ACM Reference Format:
Tarek Ramadan, AbdelRahman Abdou, Mohammad Mannan, and Amr Youssef. 2026. The Silent Spill: Measuring Sensitive Data Leaks Across Public URL Repositories. In *Proceedings of the 2026 ACM Internet Measurement Conference (IMC '26), October 12–16, 2026, Karlsruhe, Germany*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3777912.3809151

1 INTRODUCTION

Over time, URLs have evolved from simple online resource locators into carriers of sensitive data such as authentication tokens, document identifiers, and recovery links. Thus, a single leaked URL can expose an entire account, a signed contract, or an organization's internal data. Across the internet, URLs are constantly shared, scanned and stored by security analysis services, repositories, and paste sites. Some of these links may lead to private files, login tokens, password-reset pages, or personal/organizational documents. Understanding what these URLs contain and how they spread is key to seeing the real risk behind them.

Several web services, such as VirusTotal [16], Hybrid Analysis [5] and URLScan.io [14], scan user-submitted URLs for security issues, e.g., malware and phishing. Email providers and enterprise security gateways also often automatically forward (suspicious) URLs to these security scanners for analysis. These platforms often leave submitted URLs visible in public reports (e.g., for further look-up without requiring a new analysis). Other sources, such as the Wayback Machine [6] and RedHunt paste sites [10], also host and archive URLs that remain online long after use. These public URLs, however, may unintentionally expose private data. Examples include: private links revealing API tokens, and private GitHub pages and internal URLs [11]; and cloud/NAS-hosted files, corporate communication, and Oauth sign-in links [15].

Analyzing leaks from public URLs at scale presents, however, several practical challenges. URLs often expire or redirect, and public datasets from scanning platforms and web archives frequently contain redundant or benign entries. Sensitive content may also appear only after dynamic rendering, and web content exists in diverse formats—HTML, PDFs, images, and dynamically generated elements—requiring multiple extraction methods. In addition, maintaining accuracy across millions of URLs requires balancing strict filters to reduce false positives with flexible analysis to capture potential leaks.

We developed a lightweight detection system that combines lexical analysis, dynamic content inspection, and category-based sensitivity classification. Our goal is to analyze many public sources while maintaining accurate and efficient detection. We collected 6,094,475 URLs from public scanning services and web archives, then filtered and analyzed them for potential exposures. We found 12,331 potential exposures, including 26 live password-reset links, 83 API keys in query parameters or inline content, and 12 publicly accessible e-signature workflows. We also found persistent 2FA backup codes and unexpired JSON Web Token (JWT) tokens. We notified all repository operators and 53 affected parties, several of whom responded with acknowledgments and fixes.

IMC '26, October 12–16, 2026, Karlsruhe, Germany

ACM ISBN 979-8-4007-2327-8/2026/10
https://doi.org/10.1145/3777912.3809151

This paper presents our initial contributions:

- We build a cross-platform dataset of roughly six million URLs collected from five public sources.
- We design and implement a scalable and lightweight methodology[1] for detecting potential URL leaks.
- We apply our system, which uses lexical, structural, and dynamic inspection, to measure potential URL-based exposures.

2 RELATED WORK

Smaragdakis et al. [9] trained a machine-learning classifier on a billion URLs from Common Crawl [2] using Curlie.org[3] categories to categorize URLs that belong to five sensitive categories: Ethnicity, Health, Political Beliefs, Religion, and Sexual Orientation. Borders and Prakash [1] introduced a framework to measure how much information web requests reveal, ignoring predictable details such as browser headers or timestamps that are not actual leaks. West and Aviv [17] developed a tool to measure privacy disclosures in URL query parameters, showing that identifiers and credentials can be exposed through GET requests. They provided a classification of parameter types associated with private data, advancing understanding of URL-based information disclosure. Kim et al. [7] developed a phishing detection system using structural and lexical features such as token distribution, path depth, subdomain entropy, and parameter naming conventions. Their framework demonstrated how structural and lexical attributes can effectively characterize suspicious URLs. GitHound [13] and SecretFinder [8] apply regex-based pattern matching to identify secrets in GitHub repositories and JavaScript files. These tools are used in development workflows to detect API keys before release, contributing to automated secret detection practices.

Research gap. Overall, existing studies examine URL-based data exposure from different perspectives, but do not capture the current scale and nature of URL-based leaks. Smaragdakis et al. [9] performed large-scale classification of sensitive URLs. A sensitive URL and a URL leak are quite different. A URL is sensitive if it falls under any of the five categories above [9]. A URL leak provides direct public access to a private object that should not be publicly accessible. For example: `example.org/cancer-support/` is a sensitive URL because it is health related, but we do not consider this a leak herein, whereas `exa.com/invoice.pdf?acs_tkn=AF3` is a URL leak, but it would not fall under the defined sensitive categories. West and Aviv [17] analyzed query-string disclosures and built a classification of sensitive parameters, but did not cover dynamic or archived sources. Borders and Prakash [1] introduced an early framework quantifying information flow in HTTP requests, but did not target concrete data leaks. GitHound [13] and SecretFinder [8] detect exposed keys in code repositories, but do not cover live or dynamic web analysis. Despite these advances, the scale, persistence, and cross-platform nature of URL leaks remain unmeasured, motivating a systematic detection and measurement effort across security analysis, archival, and sharing platforms using dynamic rendering and large-scale inspection. We address this gap herein.

[1]Source code available at: https://github.com/Madiba-Research/silent-spill-heuristic-system

Table 1: Static-resource types excluded in Stage 1. PDFs are retained for text/OCR.

Category	Extensions (examples)	Handling
Stylesheets	`.css`	Exclude (static)
Fonts	`.woff`, `.woff2`, `.ttf`, `.otf`, `.eot`	Exclude (static)
Icons	`.ico`	Exclude (static)
Images	`.png`, `.jpg`, `.jpeg`, `.gif`, `.bmp`, `.svg`, `.webp`	Exclude (static), *except* whitelisted domains[2]
Media	`.mp3`, `.mp4`, `.mov`, `.avi`	Exclude (static)
PDFs	`.pdf`	**Retain** (text/OCR)

3 OUR FRAMEWORK

To systematically identify sensitive data exposed via public URLs, we present an automated detection system that efficiently processes large datasets and detects potential leaks embedded in HTML structure, metadata, or URL parameters by combining keyword-based filtering with content analysis across rendered pages, structured fields, and OCR extraction. Figure 1 summarizes the overall workflow. We detail each stage below.

3.1 Initial List of URLs

To build a diverse dataset of potentially sensitive URLs, we collect unique entries between January and April 2025 from five sources covering real-time analysis platforms and historical archives: VirusTotal, Hybrid Analysis, URLScan.io, Wayback Machine, and Redhunt paste sites. URLs are collected from each source as follows.

(1) *URLScan.io [14]:* We use domain- and keyword-based queries to retrieve 1,000-URL batches via the Search API,[3] contributing 5,515,585 URLs in total. (2) *VirusTotal [16]:* We query the VirusTotal API[4] across risk-prone domains (Table 5, in the appendix) frequently used for payments, authentication, and document sharing, yielding 132,919 URLs. (3) *Wayback Machine [6]:* We use the CDX API[5] to collect archived URLs, contributing 363,550 entries. Queries use prefix matching (e.g., `.example.com/`) with JSON output, retaining accessible URLs across domain groups (Table 6, in the appendix). (4) *Hybrid Analysis [5]:* We extract URLs from public sandbox reports via the open feed,[6] processing 3,000 reports-which represent all publicly accessible reports during the collection period-yielding 68,892 URLs. (5) *RedHunt Paste Sites [10]:* We scan publicly available paste-sharing domains through live scraping and archived index pages,[7] collecting 13,529 URLs (Table 7, in the appendix).

The initial dataset contains 6,094,475 URLs (see the first three columns of Table 3). This dataset serves as the input to our four-stage leak-detection system.

3.2 Stage 1: URL Filtering

We begin by filtering non-HTML static resources such as stylesheets, fonts, and icons (e.g., `.css`, `.woff`, `.ico`); however, we keep `.pdf` files, as they may contain sensitive text or embedded data. Static file types are listed in Table 1. We also retain both HTTP and HTTPS links. After removing static resources, 4,982,613 URLs remained. We

[3]https://urlscan.io/docs/api/
[4]https://developers.virustotal.com/reference/overview
[5]https://web.archive.org/cdx/search/cdx
[6]https://www.hybrid-analysis.com/search?query=
[7]https://redhuntlabs.com/online-ide-search/

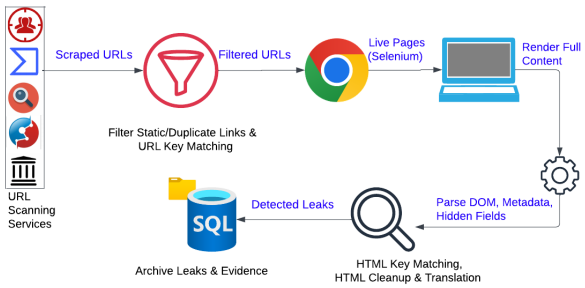

Figure 1: Leak Detection Workflow Overview

keep URLs from image-sharing, paste, and e-signature platforms regardless of file type to maintain domain coverage, and we store both screenshot and HTML copies for later analysis.

We curate 302 identifiers (Table 10 in the appendix) used for URL-parameter leak detection, aggregated from prior work [9, 17], public datasets, and live web fields. The list includes security terms (e.g., `auth`, `token`, `session`) and operational markers (e.g., `invoice`, `booking`, `tracking`) to detect both security and business-related exposures. We then perform lexical matching on two parts of each URL: (1) the path segments, and (2) the query parameters and their values. For example, in: https://www.example.com/aa/bb?var1=foo&var2=bar, the path segments are `aa`, `bb`, and the parameters are `var1= foo`, `var2=bar`. Each element is checked for substrings that match any of our 302 identifiers. Lexical matching reduced the dataset to 2,104,376 URLs.

We also retain URLs meeting any of the three structural criteria: (i) URLs containing at least five parameters or field names longer than seven characters; (ii) a total length exceeding 150 characters; and (iii) four or more directory levels, common in nested or internal resources. We set these thresholds after testing them on a 10K-URL pilot dataset, where 7,842 URLs (78.42%) showing potential leaks followed these patterns. In a manual review of 100 URLs from the retained set and 100 from the remainder, 92 retained URLs contained leaks, compared to 7 in the non-retained group. Finally, short or single-parameter URLs (e.g., `?id=123`, `?t=abc`) are retained for entropy-based analysis to detect random tokens or identifiers. This ensures concise but potentially sensitive URLs are not discarded. Applying the structural criteria further reduced the set to 832,714 URLs. Combined, Stage 1 reduces the initial 6,094,475 URLs to 832,714 candidates. Table 2 shows the reduction in URL counts after each stage.

3.3 Stage 2: Retrieve Content

Each URL passing lexical filtering undergoes live validation to confirm reachability and responsiveness. We send an HTTP HEAD request via Selenium; URLs responding with status 200–399 are marked live and proceed to a full GET request to fetch the page body. URLs that return client or server errors, timeout, or fail to respond are skipped.

To handle redirections safely, we allow up to three consecutive redirects, based on a random test of 10,000 URLs in which 8,976 (89.76%) of valid pages resolved within three hops. This threshold is chosen to ensure consistent coverage without excessive crawling. This approach captures final destinations (e.g., expanded URL shorteners) that may contain sensitive data. We exclude live URLs displaying messages such as "Access Denied," "Authentication Required," or "404 Not Found," (see Table 8, in the appendix). This ensures that only content-rich, accessible pages proceed to further analysis. Overall, this availability check eliminates ~82% of the filtered candidates, yielding 149,450 live and content-rich URLs.

Table 2: Reduction and filtering stages applied to the original dataset

Stage	Remaining	Eliminated	% Eliminated
Initial dataset	6,094,475	–	–
Filter target URLs	832,714	5,261,761	86.3%
Availability check	149,450	683,264	82.03%
Confirmed leaks	12,331	137,119	91.75%

3.4 Stage 3: Render and Extract

To capture exposures requiring client-side rendering, each candidate URL is loaded in a headless browser. The browser loads dynamically generated content, including JavaScript-injected elements, and auto-filled fields that are not available in static HTML. Client-side scripts are allowed to run, and the viewport expands to full scrollable height to ensure all rendered content is visible. This step reveals information that appears within a 5-second render window and is absent from the initial static HTML response. We chose 5 seconds as longer renders (10 seconds) revealed less than 1.96% additional leaks across a random test of 10,000 URLs. During rendering, visible non-English text is translated via the Google Translate API [4]. After rendering, we perform a structural inspection to prepare each page for detailed content analysis. We analyze the full DOM to capture JavaScript-injected elements, dynamic forms, and HTML input fields marked as `type=hidden`, which store data not visible on the rendered page. We parse embedded metadata (e.g., `data-*` attributes and meta tags) to surface tokens or document identifiers that may not appear in visible text. We also use Tesseract OCR [12] to extract text from images so that information visible only in images is also analyzed. We apply OCR or deeper inspection when a page shows very little readable text (e.g., mostly images or icons), or when the URL suggests hidden sensitive data (e.g., reset or token links). This ensures all observable textual, structural, and visual content is available for subsequent analysis.

3.5 Stage 4: Content Analysis

At this stage, we analyze page content to detect potential leaks. We sanitize the HTML by removing non-content elements such as `<script>` and `<style>`. We analyze the remaining content across three data types: visible text, input field values, and metadata attributes. We evaluate all extracted values against 402 curated identifiers (Table 9, in the appendix), extending the Stage 1 parameter set with additional context terms derived from observed leaks, public datasets, and metadata patterns. Keyword-based filtering provides the initial candidate set, and we perform structural and contextual checks to confirm if matches represent potential leaks.

4 RESULTS

We now present the leak distribution by category, and highlight our critical findings; for an overview, see Table 3 and Figure 2.

Table 3: Detected Leaks Across URL Sources

Source	# URLs	% of dataset	Leak Matches	% of Total
URLScan.io	5,515,585	90.48%	7,826	63.5%
Wayback Machine	363,550	5.97%	1,806	14.6%
VirusTotal	132,919	2.18%	1,445	11.7%
Hybrid Analysis	68,892	1.13%	963	7.8%
Redhunt paste sites	13,529	0.22%	291	2.4%
Total	6,094,475	100%	12,331	100%

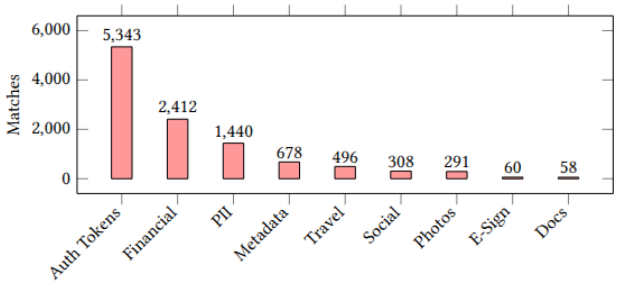

Figure 2: Distribution of Flagged Leak Categories

4.1 Leak Distribution by Category

Our system processed 6,094,475 URLs end-to-end in 3.8 hours (Ubuntu 24.04.2 LTS on x86_64, Intel Core i9-10900, Linux kernel 6.14.0, 64 GB RAM, 2 TB HDD), identifying 12,331 potential URL-based leaks, grouped into 9 categories representing the most common leak types observed in our dataset. Figure 2 shows the categories and the number of potential leaks flagged in each category. *Authentication & Tokens* leaks were the most common (5343, 43.3%), indicating that credential-related artifacts appear most frequently in the analyzed URLs. To verify that the flagged content is true positives (TPs), we chose a random sample from each category for manual inspection. Table 4 shows the sample size we took, the number of false positives (FPs) discovered upon manual inspection, the FP rate, and the Wilson score interval at 95% confidence [18]. Sample sizes differ slightly across categories because we chose numbers that kept the inspected URLs roughly similar across categories while keeping the manual review manageable. Considering the upper bound (UB) FP rate of 0.26 (26%) for the Authentication & Tokens category, the results show that the flagged potential leaks contain $5343 \times 0.74 = 3,954$ TP URL-based leaks (with 95% confidence).

The verification process followed consistent labeling rules. One author marked a URL as a true positive if it exposed sensitive data (e.g., API key, token, invoice number, or identifier), and as a false positive if it contained placeholders or benign metadata (e.g., `<meta name="author" content="Admin">`). A total of 3 ambiguous cases were reviewed by a second author, of which 2 were confirmed as true positives and 1 as a false positive.

For personally identifiable information (PII), examples included exposed email addresses, full names, or phone numbers in visible text or metadata. Social platform leaks included invitation links, chat tokens, and webhook URLs from services such as Discord or Slack. Metadata-related leaks involved internal reference tags or descriptive fields (e.g., `created_by=admin`) that revealed organizational context. Among all categories, credential-related leaks dominated. Many matched fields such as `authentication_token` (2442) and `verification_ token` (347), accounting for 43.3% of all exposures. Financial artifacts ranked second (2412, 19.6%), commonly matching parameters like `checkout_id` (1,117) and `payment_token` (313), often embedded in order confirmations or client-side payment flows lacking additional access control. Such identifiers could expose order details or enable replayed payment requests. Other categories, including document exposures and location artifacts, appeared less frequently.

Most potential leaks (63.5%) originate from URLScan.io (see the last two columns of Table 3), where public scans often reveal API and session URLs submitted for reputation analysis. The Wayback Machine contributes 14.6% of potential leaks retrieved through its CDX indexing API, which provides access to archived web snapshots. These URLs typically stem from content that was once public—such as invoices, documents, or dashboards—but remains accessible in archived form even after its intended use was completed. VirusTotal accounts for 11.7%, mainly exposing password-reset links. Hybrid Analysis contributes 7.8%, while Pastebin yields 2.4%, primarily from static credentials, configuration data, or file links shared in public pastes.

Table 4: Sample size (SS), False Positives (FP), FP Rate, and Wilson interval (Lower and Upper Bounds).

Category	SS	# FPs	FP rate	Wilson Score	
				LB	UB
Authentication & Tokens	30	3	0.10	0.035	0.26
Personally Identifiable Info	31	4	0.13	0.052	0.29
Financial Information	33	4	0.12	0.047	0.27
Document Access	32	4	0.13	0.053	0.29
Travel & Booking	30	4	0.13	0.051	0.29
Social Platforms & Messaging	34	5	0.15	0.066	0.30
Miscellaneous Metadata	30	4	0.13	0.051	0.29
E-Signature Credentials	12	0	0	0	0.24

4.2 Critical Exposure Findings

Manual validation reveals several recurring noteworthy exposure types, summarized below.

Exposed 2FA backup codes. Backup codes are static, pre-generated tokens used to bypass two-factor authentication (2FA) when users lose access to their primary method. We identified 7 rendered pages displaying fully visible codes, typically 8–10 characters long, and 16 showing partial masking (e.g., half-hidden digits). Because backup codes are rarely rotated, several may remain valid over time, creating long-term risks until affected accounts are manually resecured. Fully visible codes could allow account takeover, as long as they remain valid.

Publicly accessible password reset links. We found 26 password reset links appearing in rendered content or as query parameters. For example, one reset URL appears on a social media recovery page and another on a retail checkout subdomain (`checkout.shopify.com`), both containing one-time tokens that could allow unauthorized resets if accessed. If exploited before expiration, these reset tokens can allow unauthorized password changes, resulting in immediate account compromise.

Exposed persistent JSON web tokens (JWTs). JWTs are commonly used to maintain authenticated sessions and are expected to expire after short periods. We identified 62 URLs containing tokens that either lack expiration controls or remain valid for long durations. Inspection of token claims (`iat`, `exp`) shows multi-day expiration durations, and several tokens without an `exp` field. These URLs appear on domains such as *iris.cigna.com* and *identitydev.freedomconnect.com*, which correspond to authentication or staging environments. Such tokens can enable long-term unauthorized access even after logout.

Unprotected e-signature workflows. We identified 12 active signing sessions hosted on e-signature platforms that are publicly accessible. Examples include order contracts and job offer agreements. These URLs allow unauthorized users to view or modify contracts—posing legal and regulatory concerns by undermining the integrity and accountability of digital signing processes.

5 LIMITATIONS

First, our TP/FP estimates rely on random sampling within each leak category, assuming a uniform false-positive distribution. In practice, certain domains contributed disproportionately to FP cases (e.g., repeated login templates). Although category-level estimates remain statistically valid, domain-level skew may bias estimates. Second, while stratified manual inspection provides statistically valid confidence intervals for precision, measuring recall at scale remains difficult because no comprehensive, labeled dataset of confirmed leaks exists. Evaluating system coverage without exposing sensitive data therefore remains an open challenge. Third, source imbalance partly reflects differences in platform accessibility. URLScan.io provides broad, self-service access to historical and live results, while other repositories impose stricter access controls (e.g., approval-based access, account vetting, stricter rate limits, restricted historical search APIs, and limited bulk-export functionality). These constraints made large-scale collection uneven across sources. Although leaks were found across all platforms we sampled, future work may prioritize more balanced collection.

6 MITIGATION AND BROADER IMPACT

Our findings point to several practical steps that different stakeholders could adopt to reduce the risk of URL-based data leaks. Repository operators such as URLScan.io, VirusTotal, and archival services can apply automated filtering to redact common secret-bearing parameters (e.g., `apikey`, `sessionid`), incorporate entropy-based heuristics to hide high-entropy tokens before publication, and shift toward private-by-default policies with tiered access so that unredacted results are visible only to submitters or vetted researchers. Platforms can additionally default public reports to masked values while preserving parameter names and structural context, limiting direct credential exposure while maintaining analytical transparency. However, determining which parameters are truly sensitive is context-dependent, and automated masking may miss secrets embedded in unconventional or encoded formats. As an additional measure, repositories could introduce delayed-publication controls, allowing time to revoke unintentionally exposed sensitive links before public indexing.

Service providers and developers can strengthen application design by minimizing the use of GET parameters for credentials, employing short-lived or single-use tokens for reset and recovery links, masking identifiers in public pages, and enforcing cache-control and strict referrer policies to prevent leakage via browser logs or intermediaries. At the enterprise and end-user level, organizations can configure security tools not to submit sensitive links to public repositories, deploy internal scans to detect secrets in URLs before they leave the corporate network, and monitor public feeds for exposures of their own domains so that leaked links can be revoked promptly. While no single measure can eliminate these risks, adopting layered mitigations across repositories, providers, and users can substantially reduce the prevalence and persistence of URL-based leaks.

7 CLOSING REMARKS

We showed that web resources directly accessible through their URLs, without additional access controls, are prone to public listing on URL analysis platforms and web archives once indexed. We uncovered thousands of potential private resources accessible through direct URLs that were not intended to be public. This includes 26 live password reset links, 313 payment tokens, and 62 URLs exposing authentication tokens without expiration constraints. We hope that our findings so far highlight the importance of implementing proper access control mechanisms and avoiding blatant reliance on (perceived) obscurity of URLs in protecting sensitive resources. Although confirmed leaks account for only 0.2% of analyzed URLs, their security implications are substantial. At Internet scale, this fraction corresponds to thousands of active reset links, API keys, and contractual documents that could enable account takeover, fraud, or large-scale data exposure. Our results demonstrate that URL leakage is not a marginal phenomenon but a systemic risk: even infrequent cases accumulate into a significant threat landscape requiring proactive mitigation.

All our analysis was conducted passively on publicly accessible URLs without authentication or state changes, under the safeguards described in Appendix A. To support future development and reproducibility, we release our pipeline code and identifier sets as open source, while withholding raw datasets to avoid exposing sensitive URLs. In line with our ethical safeguards, we conducted responsible disclosure, notifying operators such as URLScan.io, the Internet Archive, and, when appropriate, individual affected organizations. Beyond measurement, our system can be repurposed defensively to help services automatically identify and redact sensitive URLs before publication. Further details on ethical safeguards are provided in Appendix A, and mitigation strategies are outlined in Section 6.

Our Efforts are Ongoing

While our current methodology successfully uncovered numerous potential leaks, our ongoing work focuses on adaptability, sustainability, and systematic evaluation of efficacy (e.g., precision and recall). To better understand temporal behavior, we performed an automated revisit of 2,000 randomly sampled URLs previously flagged as leaks to measure persistence. After 75 days, 38% (762 URL) remained publicly reachable without authentication. From this set, we manually verified a stratified subset of 200 URLs to confirm that accessible pages continued to expose sensitive content rather than placeholders or error states.

We are extending the system to better adapt to varying input types and organizational requirements. Different platforms expose data in varied formats: government services may leak structured identifiers through static fields, while commercial sites often embed sensitive data within dynamically rendered components. Detection objectives also differ by use case. For instance, enterprise security teams may prioritize accuracy and low-noise alerts for continuous monitoring, whereas analysts performing large-scale reconnaissance or threat discovery may prefer more flexible parsing and relaxed thresholds to surface weaker or unconventional signals. To support this range, we are refining our parsing logic, keyword sensitivity, and filtering strategies to adapt dynamically to source characteristics and deployment goals.

Leak patterns and exposure formats may also change over time. Web platforms update their structures, parameter names evolve, and new forms of sensitive data emerge. To maintain effectiveness, we are building longitudinal monitoring mechanisms that integrate new exposure examples, update heuristics, and expand pattern sets as needed. These updates may enable the system to keep pace with evolving exposure trends without major architectural changes. In future work, we plan longitudinal measurements to model leak lifespan, understand how quickly leaked URLs become inactive, identify platforms or time periods with higher exposure concentrations, and investigate when sensitive URLs first become exposed using timestamped collection points where available. To enable scalable and consistent evaluation, we are integrating lightweight machine learning techniques to assist in automated validation of flagged results. These models will not drive detection but may help estimate error rates, identify false-positive patterns, and prioritize likely true positives for review. This approach may strengthen our ability to quantify leak prevalence across large datasets without relying solely on sampling or manual inspection.

REFERENCES

[1] Kevin Borders and Atul Prakash. 2008. Towards quantification of network-based information leaks via HTTP. In *Proceedings of the USENIX Workshop on Hot Topics in Security (HotSec)*. USENIX, San Jose, CA, USA.

[2] Common Crawl. 2025. Common Crawl. https://commoncrawl.org. Accessed: 2025-03-03.

[3] Curlie Editors. 2022. Curlie: The Collector of URLs. https://curlie.org. Accessed: 2025-02-03.

[4] Google Cloud. 2025. Google Translate API. https://cloud.google.com/translate Accessed: 2025-02-19.

[5] Hybrid Analysis. 2024. Hybrid Analysis Threat Reports. https://www.hybrid-analysis.com/search?query=search+request. Accessed: 2025-03-03.

[6] Internet Archive. 2024. Wayback Machine CDX API. https://web.archive.org/cdx/search/cdx. Accessed: 2025-03-03.

[7] Taeri Kim, Noseong Park, Jiwon Hong, and Sang-Wook Kim. 2022. Phishing URL Detection: A Network-based Approach Robust to Evasion. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22)*. ACM, New York, NY, USA, 1769–1782. https://doi.org/10.1145/3548606.3560615

[8] M4ll0k. 2018. SecretFinder: A Tool for Finding Sensitive Data in JavaScript Files. https://github.com/m4ll0k/SecretFinder. Accessed: 2025-09-04.

[9] Srdjan Matic, Costas Iordanou, Georgios Smaragdakis, and Nikolaos Laoutaris. 2020. Identifying Sensitive URLs at Web-Scale. In *Proceedings of the ACM Internet Measurement Conference (IMC '20)*. ACM, 619–633. https://doi.org/10.1145/3419394.3423653

[10] RedHunt Labs. 2024. Online IDE Search. https://redhuntlabs.com/online-ide-search/. Accessed: 2025-03-03.

[11] Positive Security. 2022. urlscan.io Data Leaks: How Security Tools Can Leak Sensitive URLs. https://positive.security/blog/urlscan-data-leaks. Accessed: 2025-03-03.

[12] Tesseract OCR Developers. 2024. Tesseract OCR Engine. https://github.com/tesseract-ocr/tesseract.

[13] Tillson. 2019. GitHound: GitHub Sensitive Information Finder. https://github.com/tillson/git-hound. Accessed: 2025-09-04.

[14] URLScan.io. 2024. URLScan.io Web Scanner. https://urlscan.io/. Accessed: 2025-03-03.

[15] Vin01's Blog. 2024. You can not simply publicly access private secure links, can you? https://vin01.github.io/piptagole/security-tools/soar/urlscan/hybrid-analysis/data-leaks/urlscan.io/cloudflare-radar%22/2024/03/07/url-database-leaks-private-urls.html. Accessed: 2026-02-20.

[16] VirusTotal. 2024. VirusTotal Online Scanner. https://www.virustotal.com/gui/home/upload. Accessed: 2025-03-03.

[17] Andrew G. West and Adam J. Aviv. 2014. Measuring Privacy Disclosures in URL Query Strings. *IEEE Internet Computing* 18, 6 (2014), 52–59. https://doi.org/10.1109/MIC.2014.104

[18] Edwin B Wilson. 1927. Probable Inference, The Law of Succession, and Statistical Inference. *J. Amer. Statist. Assoc.* 22, 158 (1927), 209–212.

APPENDIX

A ETHICS

Assessing the extent of URL-based data leaks raises important ethical considerations. We now outline the safeguards and procedures adopted in this study.

IRB guidelines. Prior to conducting this study, we consulted with the Research Ethics Office at Concordia University to determine whether formal Institutional Review Board (IRB) review was required. We described the project scope, including the collection and analysis of URLs obtained from public URL scanning services and other publicly accessible repositories. We clarified that the study involved no interaction with human subjects, no attempts to bypass access controls, and no access to non-public systems or authenticated resources. Based on this description, the Research Ethics Office at our institution determined that formal ethics review was not required, as the study analyzes publicly accessible information and does not involve human subjects research.

Publicly accessible data. All URLs analyzed in this study originated from publicly accessible sources, including VirusTotal, the Wayback Machine, URLScan.io, Hybrid Analysis, and paste sites. All content was accessed passively via public URLs without requiring credentials or login.

Passive leak detection and classification. Our methodology remained strictly observational: we did not activate password-reset flows, use exposed API keys, or interact with any state-changing elements. For document URLs, we only performed passive `GET` retrieval of files directly exposed by the public URL, without authentication or form submission. No POST requests, credential submissions, or access-control bypasses were used. Leak classification relied solely on structural patterns, page context, token formats, and visible elements.

Validating assumptions about side effects. To minimize the possibility of unintended side effects, we verified with self-controlled

accounts that visiting reset or backup-code links did not invalidate credentials or generate new codes. To reflect the real URLs in our dataset, we enumerated the password-reset and 2FA URL patterns observed in our corpus and replicated these patterns using our own accounts before testing their behavior. We tested 24 distinct password-reset and 2FA URL patterns (covering every unique pattern identified across the 49 exposed URLs). Retrieval logs confirmed the absence of POST requests or other state-changing actions.

Limited manual inspection. Manual inspection was restricted to a small stratified subset of URLs for validation. A local password-protected interface displayed only the URL, matched value, screenshot, and HTML snapshot, enabling consistent labeling while keeping sensitive material confined to an isolated workstation. Sensitive artifacts were anonymized in all reporting.

Data storage. All collected data was stored on an encrypted, password-protected local disk with no cloud backups, physically accessible only to the first author. All data will be securely destroyed at the end of the project.

Responsible disclosure. We contacted all repository operators involved in our dataset; URLScan.io acknowledged receipt and requested the specific leak patterns (e.g., `session_id=`, `auth_key=`) to deploy fixes on their end. We also contacted 23 individual organizations responsible for identifiable exposures, of which 11 responded. Additionally, we reached out to 30 affected individuals, and 8 confirmed awareness of their reported exposures. No sensitive URLs, tokens, or user-linked data was disclosed publicly, and no external parties were granted access to the dataset.

Risk–benefit tradeoff. The incremental risk introduced by this study was minimal, in our assessment, since all analyzed URLs were already publicly exposed at the time of collection. The potential benefit is substantial: by documenting these risks and detailing appropriate safeguards, our work supports mitigation efforts by service operators and offers a reproducible framework for future research.

B SUPPLEMENTARY DATA

Table 9 contains 402 structural and semantic keywords derived from file paths, filenames, and artifacts (e.g., invoice, contract, boarding pass). Table 10 contains 302 lexical identifiers aggregated from prior work [9, 17], public datasets, and common sensitive fields (e.g., `auth_token`, `sessionid`, `apikey`). The two sets partially overlap (e.g., invoice), but one targets structural artifacts while the other targets parameter names. We include both for completeness.

Table 5: Potential High-Risk Domains Queried via VirusTotal

Payment Platforms
paypal.com, stripe.com, squareup.com, checkout.stripe.com, venmo.com, pay.google.com, appleid.apple.com, secure.adyen.com, braintreepayments.com, payoneer.com, payu.com, skrill.com, revolut.com, cash.app, zellepay.com
Social Platforms
facebook.com, messenger.com, instagram.com, whatsapp.com, linkedin.com, twitter.com, x.com, t.me, discord.com, snapchat.com, pinterest.com, reddit.com, tiktok.com, wechat.com
Cloud Storage & File Sharing
drive.google.com, docs.google.com, onedrive.live.com, dropbox.com, box.com, mega.nz, nextcloud.com, owncloud.com, icloud.com, sharepoint.com, wetransfer.com, transfer.sh, send.firefox.com
Developer Services & Code Repositories
github.com, gitlab.com, bitbucket.org, npmjs.com, pypi.org, hub.docker.com, maven.org, api.github.com, git.io, cdn.jsdelivr.net, cdnjs.cloudflare.com
Authentication Providers
accounts.google.com, appleid.apple.com, auth0.com, okta.com, onelogin.com, sso.squarespace.com, signin.aws.amazon.com, keycloak.org
Travel & Ticketing
booking.com, expedia.com, airbnb.com, tripadvisor.com, skyscanner.com, hotels.com, trainline.com, ticketmaster.com, eventbrite.com, stubhub.com, checkmytrip.com, ryanair.com, boardingpass.ryanair.com
E-Commerce Platforms
amazon.com, alibaba.com, aliexpress.com, ebay.com, shopify.com, bigcommerce.com, wix.com, woocommerce.com, etsy.com, bestbuy.com, walmart.com, flipkart.com

Table 6: Wayback Machine Leak Source Domains

Category	Domains
Document Sharing	docs.google.com, drive.google.com, app.box.com, mega.nz, pcloud.com, nextcloud.com, icloud.com, filestackapi.com, mediafire.com, transferxl.com, smash.gg, zippyshare.com, anonfiles.com, bayfiles.com, dropbox.com, send-anywhere.com
E-Signature Services	docusign.com, docusign.net, hellosign.com, dropboxsign.com, adobesign.com, secure.adobesign.com, sandbox.esignlive.com, pandadoc.com, signnow.com, eversign.com, signrequest.com, signeasy.com, onespan.com, esignlive.com, zohosign.com, assuresign.com, sertifi.com, legalesign.com, signable.com, formstack.com, signx.wondershare.com
Invoicing / Payment	stripe.com, paypal.com, squareup.com, quickbooks.intuit.com, waveapps.com, zoho.com, bill.com, xero.com, invoiceninja.com, freshbooks.com, invoicehome.com, harvestapp.com, qbo.intuit.com, my.freshbooks.com, zohoinvoice.com
Authentication	auth0.com, accounts.google.com, accounts.google.com/o/oauth2, login.microsoftonline.com, login.live.com, okta.com, auth.aws.amazon.com, passport.yandex.com, id.heroku.com, bitbucket.org, bitbucket.org/site/oauth2, gitlab.com, gitlab.com/users/sign_in, slack.com/signin, firebaseapp.com, appleid.apple.com, console.aws.amazon.com, console.cloud.google.com, login.salesforce.com, id.atlassian.com, login.cisco.com, auth.digitalocean.com
Paste Sites	pastebin.com, justpaste.it, dpaste.org, hastebin.com, controlc.com, ghostbin.com, paste.ee, rentry.co, zerobin.net, privatebin.net, pastie.io, paste.mozilla.org, termbin.com, snipplr.com, snipt.net
Media Platforms	imgur.com, flickr.com, canva.com, notion.so, figma.com, miro.com, evernote.com, youtube.com/redirect?, slideshare.net, photos.google.com
Developer / Cloud Services	github.com, gitlab.com, bitbucket.org, replit.com, glitch.me, vercel.app, netlify.app, herokuapp.com, render.com, surge.sh, firebaseio.com
Viewers / Converters	docdroid.net, pdfhost.io, scribd.com, docplayer.net, edocr.com, slideserve.com, prezi.com, slidebean.com, docsend.com, view.officeapps.live.com

Table 7: Paste and Code-Sharing Platforms (Redhunt Domains)

General Pastebins and Code Sharing Tools
paste2.org, pastebin.com, slexy.org, dpaste.org, jsfiddle.net, dotnetfiddle.net, repl.it, paste.pound-python.org, phpfiddle.org, jsbin.com, snipplr.com, hastebin.com, textsnip.com, dpaste.com, ideone.com, paste.opensuse.org, paste.lisp.org, ide.geeksforgeeks.org, play.golang.org, snipt.net, dumpz.org, bitpaste.app, codepen.io, paste.debian.net, rextester.com, paste.xinu.at, pastehtml.com, heypasteit.com, pastebin.fr, codepad.org, justpaste.it, dartpad.dartlang.org, paste.fedoraproject.org, paste.org.ru, try.ceylon-lang.org, paste.frubar.net, paste.ubuntu.com, paste.org, jsitor.com, ide.codingblocks.com

Table 8: Keywords Used for Access Denial Detection

Keyword or Phrase
access denied, you need permission, 403 forbidden, unavailable, you have been blocked, error 404, page doesn't exist, 404, unauthorized, not authorized, forbidden, restricted access, private page, invalid credentials, authentication required

Table 9: List of HTML Parameters for Leak Detection

Authentication & Session Management
envelope_id, signing_token, sessionid, session_id, session_key, sign_token, doc_hash, signature_id, access_code, signing_session, approve_token, verify_token, auth_key, esign_request, contract_id, legal_doc, agreement_id, access_token, refresh_token, auth_token, csrf, xsrf, session_token, apikey, api_key, api_secret, secret_key, client_secret, crypto_key, crypto_address, private_key, public_key, -hash, hmac, security_code, verification_code, recovery_token, reset_token, oauth_token, ouath_verifier, crypto_wallet, jwt, bearer, state_token, token, security-login, client_token, sso_token, authenticity_token, csrftoken, logincsrfparam, requestverificationtoken, security_token
User Identity, Social Media & Personal Identifiers
national_id, social_security_number, driver_license_number, tax_id, pan_number, pan_card, identity_number, ssn_last4, health_id, voter_id, employee_id, citizen_id, residence_permit, ssn_full, student_id, military_id, beneficiary_id, email_address, phone_number, mobile_number, fax_number, emergency_contact
Payment Information & Transaction Identifiers
cart_id, order_number, invoice_number, purchase_id, transaction id, transaction_token, checkout_token, order_reference, payment_ref, payment_token, payment_id, payment_method, debitcard, creditcard, cvv, bank_account, stripe_token, card_number, transaction_id, invoice_id, receipt_number, account_number, license_key, Invoice Number, Invoice ID, Invoice #, Invoice Total, Invoice Amount, Order Number, Order ID, Order #, Order Total, Order Amount, Payment Method, Payment Type, Payment ID, Payment Reference, Card Number, Credit Card Number, Card Ending, Cardholder Name, Card Type, Last 4 Digits
File Links, Document Access & Repository Tokens
fileid, attachment_id, docid, doc_token, document_id, download_token, gdrive_file_id, gdrive_share_link, dropbox_link, onedrive_link, s3_bucket, repository_url, pdf_file, docx_file, xlsx_file, gdrive, onedrive/, Download Link, Document ID, GDrive Link, Dropbox Link
Travel, Booking & Ticketing References
ticket_id, reservation_number, booking_reference, confirmation_number, flight_number, itinerary_id, passport_number, trip_id, e_ticket_number, booking_ref, Boarding Pass, Booking Reference, Reservation Number, Flight Number, Confirmation Number, E-Ticket Number, Itinerary ID, Passport Number, Trip ID
Social & Messaging Integrations
invite_link, discord_invite, discord_bot_token, slack_webhook, slack_bot_token, telegram_bot_token, telegram_chat_id, social_user_id, whatsapp_link, facebook_id, messenger_thread_id, chat.whatsapp, instagram_id, snapchat_id, skype_id, Discord Invite, Slack Invite, Telegram Bot, Messenger Thread, WhatsApp Link, Group ID, Profile ID
Address, Contact Details & Personal Information
billing_address, shipping_address, mailing_address, home_address, postal_code, full_name, first_name, last_name, middle_name, date_of_birth, place_of_birth, nationality, marital_status
Medical, Insurance & Health Records
insurance_number, medical_record_id, health_insurance_id, vaccination_id, blood_type
Shipping, Tracking & Logistics Identifiers
tracking_id, tracking_number, survey_id, event_id, shipment_id, parcel_id, order_tracking_id, delivery_note_number, waybill_number, dispatch_id
Corporate, Internal & Vendor References
employee_number, staff_id, internal_reference, project_id, vendor_id, supplier_id, purchase_order_id, contract_number, rfq_id, invoice_reference
Cryptocurrency Wallets & Blockchain Credentials
Wallet Address, Crypto Token, Transaction Hash, Mnemonic, Seed Phrase, crypto_key, crypto_address, crypto_wallet

Table 10: URL Parameters Used for Leak Detection

Category	Parameters
Authentication & Session Management	envelope_id,signing_token,sessionid,session_id,session_key,sign_token,doc_hash,signature_id, access_code,signing_session,approve_token,verify_token,auth_key,esign_request,contract_id, legal_doc,agreement_id,access_token,refresh_token,auth_token,csrf,xsrf,session_token,apikey, api_key,api_secret,secret_key,client_secret,verification,verify,private_key,public_key,hash,hmac, security_code,verification_code,recovery_token,reset_token,oauth_token,oauth_verifier,jwt, bearer,state_token,token,security_login,client_token,sid,ssid,sso_token,authenticity_token, csrftoken,logincsrfparam,requestverificationtoken,security_token
API Keys, Webhooks & Developer Credentials	webhook_url,webhook_token,slack_webhook,slack_token,slack_bot_token,discord_invite, discord_bot_token,telegram_bot_token,telegram_chat_id,github_token,gitlab_token, jenkins_token,circleci_token,ci_build_token,build_hook,api_webhook,firebase_token,twilio_sid, twilio_token,zendesk_token,zoom_jwt,notion_token,airtable_api_key,api_url
Payment Information & Transaction Identifiers	cart_id,checkout,order_number,invoice_number,purchase_id,transaction_id,transaction_token, checkout_token,order_reference,payment_ref,payment_token,payment_id,payment_method, stripe_token,paypal_token,square_token,debitcard,creditcard,cvv,cvc,bank_account,iban, bic,card_number,account_number,invoice_id,receipt_number,license_key,discount_code, promo_code,coupon_code,voucher_code,voucher_id,giftcard_number,loyalty_card_id, reward_points,balance_amount,billing
File Links, Document Access & Repository Tokens	file,file_url,fileid,attachment,attachment_id,doc,docid,doc_token,document,document_id, download_id,download_token,gdrive_file_id,gdrive_share_link,dropbox_link,onedrive_link, s3_bucket,repository_url,pdf_file,docx_file,xlsx_file,gdrive,onedrive,backup_file,database_dump, config_file,logfile,snapshot_id,backup,config
Travel, Booking & Ticketing References	boardingpass,visa,pasabordo,ticket,ticket_id,reservation_number,booking_reference, confirmation_number,flight_number,itinerary_id,passport_number,trip_id,e_ticket_number, booking_ref,travel_doc_id,miles_account_id,frequent_flyer_number
User Identity, Social Media & Personal Identifiers	invite_link,whatsapp_link,messenger_thread_id,facebook_id,instagram_id,snapchat_id, skype_id,social_user_id,group_id,chat_token,customer_id,client_id,user_id,account_id, member_id,profile_id,aadhaar_number,nhs_number,citizen_id,residence_permit, ssn_full,student_id,military_id,beneficiary_id,national_id,social_security_number,ssn, driver_license_number,tax_id,pan_number,pan_card,identity_number,ssn_last4,health_id, voter_id,employee_id,member_number,email_address,phone_number,mobile_number, fax_number,emergency_contact
Address, Contact Details & Personal Information	billing_address,shipping_address,mailing_address,home_address,postal_code,zip_code, full_name,first_name,last_name,middle_name,dob,date_of_birth,place_of_birth,gender, nationality,marital_status,address
Medical, Insurance & Health Records	insurance_number,medical_record_id,health_insurance_id,vaccination_id,blood_type, healthcare_id,dental_record_id,employer_insurance_id
Shipping, Tracking & Logistics Identifiers	tracking,tracking_id,tracking_number,shipment_id,parcel_id,order_tracking_id, delivery_note_number,waybill_number,dispatch_id
Corporate, Internal & Vendor References	employee_number,staff_id,internal_reference,project_id,vendor_id,supplier_id, purchase_order_id,contract_number,rfq_id,invoice_reference
Cryptocurrency Wallets & Blockchain Credentials	wallet,wallet_address,crypto_token,transaction_hash,eth_address,btc_address, mnemonic_phrase,seed_phrase,keystore,crypto_api_key,crypto_secret_key
Security, Risk & Audit Identifiers	audit_log_id,incident_id,security_alert_id,fraud_case_id,risk_score,compliance_report_id
Generic, Broad & Common Sensitive Terms	file_name,filename,filepath,doc_link,dataset,shared_link,public_link,direct_link,temp_link,auth, signin,login,creds,credentials,password,pwd,passwd,identity,key,certificate,ssh_key,oauth,profile, account,settings,media,media_id,photo,image,img_url,video_url,link
E-Signature Service Domains	esignlive.com,sandbox.esignlive.com,docusign.net,docusign.com,secure.adobesign.com, adobesign.com,hellosign.com,onespan.com,signnow.com,pandadoc.com,dropboxsign.com, rightsignature.com,zohosign.com,signrequest.com,eversign.com,assuresign.com,formstack.com, signeasy.com,sertifi.com,signable.com,legalesign.com,esignly.com,signx.wondershare.com, docsketch.com,getaccept.com,signaturit.com

Continued on next page

Category	Parameters
Hosting Domains	photos.google.com,lh3.googleusercontent.com,drive.google.com,dropboxusercontent.com, imgur.com,i.imgur.com,i.redd.it,preview.redd.it,cdn.discordapp.com,fbcdn.net, scontent.xx.fbcdn.net,telegra.ph,cdn4.telegram-cdn.org,mmg.whatsapp.net,onedrive.live.com, 1drv.ms,s3.amazonaws.com,bucket.s3.amazonaws.com,flickr.com,staticflickr.com, wetransfer.com,transfer.sh,user-images.githubusercontent.com,prnt.sc,snag.gy,gyazo.com, mail-attachment.googleusercontent.com,attachments.office.net,tinypic.com,imageshack.us, postimg.cc,ibb.co,freeimage.host,imagevenue.com,pixhost.to
Paste / Code Sharing Domains	paste2.org,jsbin.com,play.golang.org,paste.debian.net,pastehtml.com,pastebin.com,snipplr.com, snipt.net,heypasteit.com,pastebin.fr,slexy.org,hastebin.com,dumpz.org,codepad.org, jsitor.com,dpaste.org,textsnip.com,bitpaste.app,justpaste.it,jsfiddle.net,dpaste.com,codepen.io, dartpad.dartlang.org,ide.codingblocks.com,dotnetfiddle.net,ideone.com,paste.fedoraproject.org, paste.frubar.net,repl.it,paste.opensuse.org,rextester.com,paste.org.ru,paste.ubuntu.com, paste.pound-python.org,paste.lisp.org,paste.xinu.at,try.ceylon-lang.org,paste.org,phpfiddle.org, ide.geeksforgeeks.org