Short Path to Phishing: Identifying Misused URL Shortening Services in the Wild

Zul Odgerel*§, Yevheniya Nosyk§, Jan Bayer§, Sourena Maroofi§,
Louis Bedeschi§, Andrzej Duda*§, Maciej Korczyński*§

*Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, France
firstname.lastname@univ-grenoble-alpes.fr

§KOR Labs Cybersecurity, France
firstname.lastname@korlabs.io

***Abstract*—URL shortening services are commonly used to share long links, which avoids the limits on the number of characters imposed by online platforms. However, cybercriminals exploit these services to obscure link destinations, bypass security filters, and deceive users. Consequently, short URLs involved in phishing appear on popular blocklists, which may trigger abuse notifications to registrars or TLD registries. This misattribution forces them into manual investigations and consumes valuable time on abuse that is not under their direct responsibility. If the role of a domain as a shortening service is not recognized, it risks mistaken suspension despite that most links are benign. We argue that addressing such abuse requires tailored mitigation strategies and that maintaining an accurate and up-to-date list of URL shortening services is essential.**

In this paper, we propose a classification model to determine if a given domain name belongs to a URL shortening service. We manually curate a ground truth dataset of 211 URL shorteners and collect three groups of features to further train two machine learning models. Our random forest classifier achieves a precision of 98.4%. Next, we apply our method to 1.5 M unlabeled phishing URLs reported to APWG, OpenPhish, and PhishTank. Our model identifies 177 new USS in the wild, not previously seen in our ground truth. Finally, we measure the post-detection uptime of malicious short links from the ten most abused USS, showing that the median mitigation time is within 48 hours.

I. Introduction

This paper concerns URL Shortening Services (USS) that are commonly used to share long links. Uniform Resource Locators (URLs) are the cornerstone of the modern Web—they define the identifiers for accessing content on the Internet. Yet, despite their definition as the "compact string representation" [1], they may grow in size due to long domain names and multiple URL parameters. Apart from being difficult to recall, long URLs can reach the length limits of the Short Message Service (160 characters [2]) or online social networks such as X (280 characters [3]).

TinyURL, the first URL shortening service, was launched in 2002 with the goal of generating shorter representations (e.g., `https://tinyurl.com/drp982ny`) of long links (e.g., `https://very-long-domain-name.com/?with=many&different=parameters`) [4]. As demand grew, hundreds of similar services appeared online [5], some generating more than 600 million short URLs per month [6]. In addition to their original functionality, USS have evolved into powerful marketing tools for analytics, tracking, and monitoring [7], [8].

The widespread adoption of USS has also attracted malicious actors who exploit them for cyberattacks [9], [10]. Short URLs conceal the true destination (unless a user explicitly previews the link), reducing transparency and making it easier to mislead users. This inherent opacity is widely abused in online fraud, phishing, and malware campaigns [9], [10], [11], [12], in which attackers use short links to bypass URL-based security filters, evade blocklist-based detection, and increase click-through rates. Once clicked on, the links can redirect users to credential harvesting sites (i.e., deceptive landing pages) or to servers hosting malware payloads. As shortening services often rely on automated and anonymous usage, it is difficult to trace the origin of abuse or to enforce effective preventive controls [11], [13]. Moreover, the legitimate popularity and high traffic of shortening services can lend an appearance of trustworthiness, further increasing the effectiveness of malicious campaigns.

Short URLs used in phishing campaigns appear on popular blocklists, which may trigger notifications to registrars or TLD registries. Such cases require further assessment, despite the abuse originating from the misuse of URL shortening services rather than any malicious use of domains under their management. When the role of a given domain as a shortening service is not recognized, there is a risk of mistakenly suspending the entire domain even though most of the links it hosts are benign. For example, we observed a benign shortening service under a European country-code TLD (ccTLD) that was temporarily suspended, disrupting access for legitimate users and services (cf. Section II-B).

A more appropriate and targeted response is to notify the shortening service provider to disable the specific short link, thereby cutting off access to the malicious destination without affecting unrelated content. Once the link is deactivated, further investigation can determine whether the malicious landing page is hosted on a compromised site or a domain registered for abuse [14]. In such cases, additional notifications to registrars, TLD registries, or hosting providers may be necessary to ensure complete removal of harmful content and

979-8-3315-8969-1/25/$31.00

suspension of malicious domain names. However, given their role in managing redirection infrastructure, URL shortening service providers are particularly well-positioned to promptly mitigate abuse by deactivating malicious links.

Previous work considered the use of USS in various malicious activities [15], [9], but the studies relied on crowdsourcing or on a seed of known URL shortener domains for their analysis. While individual efforts exist to compile such lists [16], [17], their underlying methodology is often unclear, and they are typically not systematically maintained. Keeping them up to date manually is challenging, especially because dedicated software makes it easy to create new URL shortening services [18], [19].

In this paper, we propose a methodology for automated detection of domain names used by URL shortening services. It consists of training a machine learning model on a ground truth dataset of confirmed URL shortening service domains and using it to classify domain names according to three types of features: i) cluster-based, ii) lexical, and iii) redirection-based. We apply the proposed model on real-world phishing datasets collected from APWG, OpenPhish, and PhishTank to identify the domains of URL shortening services used in phishing campaigns. We analyze the most frequently abused URL shortening services and measure the uptimes of the associated malicious URLs.

To the best of our knowledge, no prior work explored the possibility of an automated method for determining whether a given domain name belongs to a URL shortening service. The results of the paper not only advance the ability to accurately identify URL shortening services at scale but also lay the groundwork for improving abuse mitigation strategies. By enabling automated classification of USS domains, this work facilitates more targeted interventions to reduce malicious activity without unnecessarily impacting legitimate services. Overall, our contributions are as follows:

- We curate a ground truth dataset of 211 confirmed generic URL shortening service domains and collect three groups of features: lexical, cluster-based, and redirection-based.
- We train a logistic regression and a random forest machine learning models to detect whether a given domain is a USS, the latter attaining a precision of 98.4%.
- We apply our method to 312 k domain names found in 1.5 M URLs blocklisted by APWG, OpenPhish, and PhishTank, identifying 177 URL shorteners in the wild to complement our ground truth dataset.
- We analyze malicious short links from the ten most abused USS, accounting for 70% of observed short phishing URLs. We find a median mitigation time of 48 hours.

The rest of this paper is organized as follows. Section II gives the necessary background on the operation of URL shortening services while Section III describes our methodology. In section IV we build and analyze two classifiers that we apply on unlabeled data in Section V. We review the related work in Section VI. The limitations and ethical considerations of this work are discussed in Sections VII and VIII, respectively. Finally, Section IX concludes the paper.

II. Background

This section provides an overview of URL shortening services to contextualize the challenges addressed in this work. We first discuss service models to show how generic shorteners differ from dedicated ones and why their openness makes them vulnerable to abuse. We then outline the threat model, define relevant actors, and describe redirection techniques commonly employed by services. Finally, we examine short link lifetimes to explain why links may become inactive due to abuse takedowns or service discontinuations.

A. Modes of Operation

Originally created as online services, modern URL shortener providers offer great flexibility to their customers. In the following, we discuss the two most common service models used in the industry:

1) Generic: These services typically offer a web interface for creating short links, with or without user authentication. The generated URLs share a common domain name (e.g., **`bit.ly`**). Users are often proposed a freemium model: basic functionality is available at no cost, while advanced features such as analytics, dedicated customer support, or higher link quotas require a paid subscription. Additionally, open-source software (e.g., YOURLS [19], Polr [18], Shlink [20]) allows anyone to deploy and operate their own generic URL shortening service.

2) Dedicated: Used exclusively by a company or a brand to share their web content and monitor click traffic, these services operate under a domain name that closely resembles the brand itself (e.g., **`bbc.in`** used exclusively for BBC articles and media) to convey trustworthiness to users. The link shortening functionality may be developed in-house or provided by a third-party USS. In the latter case, the domain's **`CNAME`** or **`A`** record typically points to the link resolution servers of the commercial USS [5]. For example, The New York Times uses the **`nyti.ms`** domain for its short links, whose **`A`** record resolves to a Bitly-operated server [21].

This paper focuses on generic USS that are publicly accessible and allow anyone to create and distribute short URLs. Dedicated shortening services are explicitly excluded from our analysis due to their restricted access, limited potential for abuse, and the lack of visibility into their internal operations.

B. Threat Model

In this work, we consider adversaries who exploit generic URL shortening services to conceal malicious destination links and distribute harmful content, such as phishing pages, malware payloads, and scam websites. Attackers leverage the anonymity and ease of use provided by USS, possibly creating accounts with disposable email addresses or using services that do not require authentication. By embedding short links in emails, social media posts, or SMS messages, they aim to bypass URL-based security filters and deceive users into clicking.

The primary victims in this model are end users who risk exposure to malicious landing pages, and USS operators, whose domains may suffer reputation damage or face suspension by registrars or TLD registries if abuse appears widespread or the domain role as a shortening service is not recognized. Defensive stakeholders include USS operators, domain registrars, hosting providers, and security vendors maintaining blocklists.

The key implication of our threat model is that abuse notifications should ideally be sent first to the USS operators, allowing them to deactivate specific malicious short URLs and prevent further harm without disrupting the entire service. In parallel, separate notifications regarding the landing pages, i.e., the final destinations to which short URLs redirect, should be sent to other relevant intermediaries, such as registrars for maliciously registered domains or hosting providers for compromised websites [22], [23]. This tiered approach enables each actor in the ecosystem to respond appropriately within their scope of control, reduces unnecessary workload for registrar and registry abuse desks, and minimizes the risk of collateral damage through domain-level suspensions.

As an illustration, our measurements revealed that a benign generic shortening service under a European ccTLD was temporarily suspended at the registry level (evidenced by the **`serverHold`** EPP status code, and the DNS **`NXDOMAIN`** response for the given domain name) in late 2024, lasting approximately 24 hours. Although we observed malicious short URLs on this service during this period, the reasons for the suspension remain anecdotal. This case nevertheless highlights the potential for disruption to legitimate services and underscores the importance of more targeted mitigation strategies.

This threat model motivates our work: the ability to identify generic USS domains automatically leads to more accurate domain classification and enables security teams to differentiate between malicious landing domains and intermediary USS domains. Such differentiation is critical for enabling proportional responses and avoiding unnecessary disruption of legitimate services. We explicitly exclude attackers who compromise the USS infrastructure itself or abuse dedicated URL shorteners with restricted access, as such scenarios are beyond the scope of this study.

C. Redirection Techniques

Link resolution servers are key components of URL shortening services—they redirect the user to the target URL. Understanding the redirection mechanisms is important for our classification methodology, as they help distinguish the USS domains from others. In the following, we discuss three redirection techniques commonly used by URL shortening services:

1) Server-Side Redirection: When the user requests a short URL, the link resolution server responds with an HTTP redirection status code (in the form of **`3xx`**) and includes the destination in the HTTP **`Location`** header. The client browser then automatically redirects to the target URL before rendering any content. This redirection happens almost instantly, resulting in a minimal delay for the user. The example below illustrates how Bitly (**`bit.ly`**) uses the **`301`** HTTP status code (**`Moved Permanently`**) to redirect from **`https://bit.ly/3Bg19uM`** to its final destination:

```
HTTP/2 301
server: nginx
...
location: https://www.usatoday.com/story/travel
   /2022/02/10/amtrak-deal-valentines-offer-sale
   /6741296001/
...
```

2) HTML Redirection: This client-side redirection technique operates by setting the **`<meta>`** tag in the **`<head>`** section of the HTML page. Specifically, the **`http-equiv`** attribute is set to **`Refresh`** and the **`content`** attribute specifies the number of seconds to wait before the redirection, followed by the target URL. Below is an example of short URL **`https://cutt.us/94oyf`**:

```
<meta http-equiv="refresh" content="0;URL=https://
   www.mathworks.com/help/sps/ref/solarcell.html">
```

The browser loads the response body, executes any embedded scripts, and then waits for the specified amount of time before performing the redirection. Some USS use this technique to briefly display the destination URL or an advertisement. The HTML **`<meta>`** tag simulates this behavior by replicating the effect of the corresponding HTTP **`Refresh`** header. Notably, the same functionality can be achieved directly via the HTTP **`Refresh`** header, where both the delay and the destination URL are specified. Despite being part of the HTTP response, the browser HTML rendering engine processes this header in the same way as the **`<meta> Refresh`** tag.

3) JavaScript Redirection: Another client-side method is triggered after the web page is partially or fully loaded, executing the JavaScript code embedded in HTML. This approach is often preferred for conditional redirects. The browser window is exposed to JavaScript through the global **`window`** object, while the HTML document loaded within that window is represented by the **`document`** object. Redirection is typically achieved by modifying the **`location`** property of either of these objects. There is no standardized way to perform JavaScript redirection. Below is an example of a Lav Linket Kortere short URL (**`https://llk.dk/0ifvzr`**):

```
<script>
   setTimeout(function(){
     window.location.href =
     'https://machinelearningmastery.com/
     an-introduction-to-recurrent-neural-networks
     -and-the-math-that-powers-them/';
     }, 15000);
</script>
```

URL shortening services mostly use server-side redirections, although other methods are also implemented, especially by ad-based shortening services [24]. In Section III, we describe how we detect each of these redirection methods.

D. Short Link Lifetimes

Short URLs may become unavailable due to expiration, abuse-related takedowns, or service shutdowns. A short URL is considered inactive when it no longer redirects to its originally intended target. Below we discuss the main reasons for which short links may stop working.

1) Link Expiration: Many URL shortening services guarantee that short links never expire [25], [26]. However, some offer features that allow link creators to define custom lifetimes, typically reserved for paying customers. This functionality is particularly useful for time-limited marketing campaigns or events with a restricted number of participants. For example, Rebrandly allows its professional plan subscribers to set expiration dates [27], while the Short.io Pro customers can limit the maximum number of times a short link can be clicked on [28].

2) Abuse and Takedown: Short URLs abused to distribute phishing pages, malware, or other malicious content are typically taken down quickly. Popular USS rely on blocklists, threat intelligence feeds, and user reports to identify harmful links and deactivate them. Some services acknowledge their limited ability to control user-generated content, stating that "since short links are created by individual users, we are unable to control the content of every short link created" [29].

Most providers offer abuse report forms on their websites, allowing anyone to flag a suspicious short URL. For example, Buffer encourages users to report harmful content not only to the shortening service but also to the social media platforms sharing the link [30]. Depending on the provider, takedowns may be handled manually by abuse teams or automatically through detection systems.

In rare but severe cases, the entire domain of USS may be suspended if they are overwhelmed by abuse or fail to respond to complaints. For instance, **`tny.im`** was forced to shut down after its hosting provider, overburdened with abuse notifications, required the service to move elsewhere [31]. However, such measures are generally avoided because they disrupt legitimate users alongside malicious ones.

3) Discontinued Service: End users also face risks when URL shortening services cease operations entirely. Sniply, for example, anticipates that if it shuts down, all its existing short links will remain active for at least five years [32], giving customers time to migrate.

III. Methodology

This section outlines our methodology on building the ground truth dataset for further use in classification. Having analyzed a considerable number of short URLs, we determined two design principles shared by popular USS: URL syntax similarity and external redirections. We start by presenting an algorithm to cluster URLs under the same domain based on their syntax structure similarity. Then, we describe our approach to capturing URL redirection chains. Finally, we collect URLs for selected candidate USS/non-USS domain names.

A. Pattern Tree-Based URL Clustering

The URL syntax structure is hierarchically organized from left to right, making it easy to be represented as a pattern tree. For each URL in our dataset, we decompose the **`path`** components into multiple tokens using the **`/`** delimiter. Figure 1 shows an example of such a tree built for URLs under **`https://skr.sk`**. Since all the entries share the same hostname, **`https://skr.sk`** becomes the root node of the tree. We then add one URL at a time creating two nodes for each token: the exact match (**`/contact`**) and the regex representation (**`/[a-z]{7}`**). These nodes are therefore the children of the previous token. Once the tree is built, there exists a pattern path for each URL in the dataset with the URL itself being in the leaf node. We form clusters by keeping all the patterns with at least five URLs.

B. Building Redirection Chains

We expect URL shortening services to exhibit some form of external redirection for their short URLs. We refer to the redirection methods described in Section II-C (Server-side, HTML, and JavaScript redirections). Server-side or HTML techniques are straightforward to detect with or without a full-featured browser. We therefore opted for a non-browser based approach using the Python's requests library [33]. On the other hand, detecting JavaScript redirection is more difficult because of its dynamic nature, so we chose the Playwright framework for web automation [34]. Below, we describe how we detect the redirections:

1) Server-Side Redirection: We capture the HTTP status code of the server's response. If it is among **`301`** (**`Moved Permanently`**), **`302`** (**`Found`**), **`303`** (**`See Other`**), **`307`** (**`Temporary Redirect`**), **`308`** (**`Permanent Redirect`**) and the **`Location`** header is set to a URL, it indicates redirection. We also check the target to ensure that we do not classify canonical redirection (non-www to www transition or protocol upgrade from HTTP to HTTPS) as meaningful redirection.

2) HTML Redirection: We look at the **`<meta>`** tag in the **`<head>`** section of the HTML page or the **`Refresh`** HTTP header.

3) JavaScript Redirection: We log the client browser navigation using Playwright. JavaScript redirection may theoretically result in cloaking but we assume that a legitimate URL shortening service would have no reason to do so. Most USS would redirect without the need for user interaction. While there may be some USS that show intermediate landing pages redirecting upon a user click, our technique fails to detect this case. Nevertheless, we log the automatically loaded redirection.

Finally, we attempt to capture redirection targets. We distinguish redirections as external or internal, depending on the target domain name. If the registered domain name of the destination is the same as in the original URL, we consider it as an internal redirection. Otherwise, we define it as external. We assume that generic USS URLs have a large number

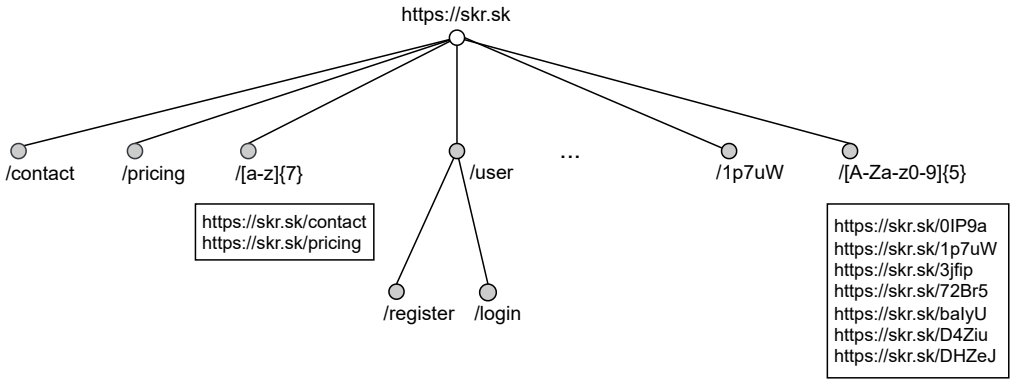

Fig. 1. Example pattern tree built for URLs under **`https://skr.sk`**. There is only one valid cluster (we set minimum cluster size to 5) formed at the rightmost node.

of external redirections to various target websites hosted on different domains.

C. URL Collection

Given a domain name, we need to obtain the associated URLs to build clusters. We refer to the Wayback Machine—a digital archive of the World Wide Web crawled by the Internet Archive [35]. This resource is known to serve the snapshots of crawled URLs of domain names in the selected date range. Its CDX API makes it possible to automate the data access [36]. We retrieve URLs archived in the past five years for each domain in our dataset.

Next, we remove the **`query`** part (as it potentially contains sensitive data) before storing the results locally. We confirm that the **`query`** part is not used as the unique identifier except for one USS in our list. We then process the URL components to identify links that point to the same resource, therefore deduplicating our dataset. Below are four URLs normalized to a single representation of **`https://shorturl.at/XIPKB`**:

```
http://www.shorturl.at/XIPKB
https://www.shorturl.at/XIPKB
https://www.shorturl.at:443/XIPKB
https://shorturl.at/XIPKB
```

We also note that certain URLs from the Wayback Machine are percent-encoded, e.g., **`*`** is represented as **`%2A`** to ensure that all the characters belong to the ASCII set, as described in RFC 1738 [1]. Since such encoding changes the syntax patterns and the lengths of URL components, we decode them. We then remove URLs pointing to non-HTML file resources such as **`*.jpg`**, **`*.js`**, **`*.php`**, **`*.pdf`**, etc.

D. Ground Truth Dataset

Binary classification requires a positive class, which in our case corresponds to the domains hosting USS, and a negative class, the services other than USS.

1) Compiling Candidate USS Domains: We start by curating an initial list of 1,579 known URL shortening service domains using two public datasets [37], [17]. We then perform a DNS scan to select the domains returning the **`NOERROR`** response code, thus only keeping the active ones. This refined list contains 1,567 entries. Next, we remove the dedicated URL shortener domains (cf. Section II-A2). To identify such services, we focus on top commercial USS providers, namely Bitly, Shortio, and Rebrandly. We perform another DNS scan of the **`A`** and **`CNAME`** resource records, matching them against the public list of IP addresses of the aforementioned services [21], [38], [39], [40]. We remove the 750 dedicated URL shortener domains. We manually verify all the remaining entries in our list of the inferred URL shorteners. We look for the evidence that a given domain name belongs to a generic URL shortening service by visiting the corresponding website. We then attempt to shorten a long URL and follow the short URL redirection to confirm the functionality. We exclude 425 more domains at this stage, as they are self-hosted dedicated USS (contrary to being hosted by popular commercial providers identified previously) or no longer provide the URL shortening service. Our candidate USS list contains 261 domains serving both commercial and self-hosted generic USSs.

2) Compiling Candidate non-USS Domains: We leverage phishing blocklists and the Tranco popularity list [41] to collect candidate domains for the negative class. We start by sampling 260 domains at random from the phishing dataset presented in Section V-A. We select phishing domains because the classifier will ultimately be applied to phishing URLs to support more effective mitigation. Moreover, phishing URLs may exhibit similar behavior to USS such as redirecting entry URLs to phishing landing pages. Likewise, we visit the websites of all the domains in the negative class to ensure they are not URL shortening services, which results in 258 domains.

We then randomly sample 1,000 domains from the Tranco top 1M list to represent benign domains. Tranco includes popular and widely used websites, making it a good baseline for non-malicious behavior. A larger sample was chosen since most legitimate services are not expected to form URL clusters. After removing three USS already labeled in the

TABLE I
THREE GROUPS OF FEATURES COLLECTED FOR ALL THE DOMAIN NAMES IN THE GROUND TRUTH DATASET. LEXICAL AND REDIRECTION FEATURES ARE EXTRACTED FROM URLS IN THE BIGGEST CLUSTER ONLY. MOST OF THE LEXICAL FEATURES HAVE THE MEAN AND THE STANDARD DEVIATION VARIANTS.

	Feature	Description
Cluster-based	*Nb_urls_clustered*	Total number of URLs that form clusters
	Avg_cluster_size	Average number of URLs in clusters
	Biggest_cluster_size	Number of URLs in the biggest cluster
Lexical	*Path_length*	URL path character length (uniform in a cluster)
	Digit_ratio_mean	Digit-to-path-length ratio of URLs (mean)
	Digit_ratio_std	Digit-to-path-length ratio of URLs (standard deviation)
	Vowel_ratio_mean	Vowel-to-path-length ratio of URLs (mean)
	Vowel_ratio_std	Vowel-to-path-length ratio of URLs (standard deviation)
	Uppercase_ratio_mean	Uppercase letter-to-path-length ratio of URLs (mean)
	Uppercase_ratio_std	Uppercase letter-to-path-length ratio of URLs (standard deviation)
	Bigram_score_ratio_mean	Character bigram score of URLs (mean) to path length ratio
	Bigram_score_ratio_std	Character bigram score of URLs (standard deviation) to path length ratio
	Hamming_distance_ratio_mean	Pair-wise URL hamming distance-to-path length ratio (mean)
	Hamming_distance_ratio_std	Pair-wise URL hamming distance-to-path length ratio (standard deviation)
Redirection-based	*Nb_redirecting_urls*	Number of URLs in the cluster redirecting to external domains
	Nb_target_domain	Number of unique destination domains of URLs
	Redirecting_url_ratio	Ratio of redirecting URLs to all the URLs

previous step, we visit the websites of the domains to ensure that they offer legitimate services other than URL shortening. Accurately labeling websites as malicious or benign is inherently challenging. In cases for which a domain is suspected of malicious activity or lacks sufficient content for reliable classification, it is excluded from the dataset. This filtering process results in the final set of 647 domains labeled as benign.

3) Removing Unqualified Candidates: We then collect URLs for each domain in the list from the Wayback Machine as described in Section III-C and obtain an intermediary dataset of 647,598 URLs from all 1,166 domains. Subsequently, we sample 100 URLs per domain name—this threshold is set to obtain a sufficient number of samples for the clustering step. After removing 138 entries (36 USS, 102 non-USS) for which the number of crawled URLs in the Wayback Machine does not reach 100, we keep the dataset of 102,800 URLs across 1,028 domain names. We then cluster the candidate domain URLs using the pattern tree, which results in the removal of 362 non-USS domains that do not form a cluster. As expected, URLs from the majority of non-USS websites do not follow the same syntactic pattern.

Lastly, we build redirection chains for the clustered URLs. We observe that in some cases all the URLs return the HTTP client error (`4xx`), the server error (`5xx`), or other HTTP connection errors due to several factors, including service unavailability at the time of the measurement, inactive or expired URLs, or the presence of CAPTCHA challenges preventing automated access. We exclude these domains from further analysis as we cannot capture the redirection chains. Overall, our ground truth dataset contains 211 USS domains and 491 non-USS domains (233 from the Tranco list and 258 from phishing blocklists).

IV. CLASSIFICATION

This section describes how we transform the above-collected ground truth dataset into a collection of features. We then train and evaluate two machine learning models used to determine whether a given domain name belongs to a URL shortening service or not.

A. Feature Engineering

We transform the previously collected dataset into a set of 17 features across three categories: cluster-based, lexical, and redirection-based. Table I summarizes them all:

1) Cluster-Based: We expect USS domains to form very few but large URL clusters. In particular, we log the number of URLs that form clusters (*Nb_urls_clustered*), the average size of all the URL clusters (*Avg_cluster_size*), and the number of URLs in the biggest cluster (*Biggest_cluster_size*). For example, a domain name with all the 100 URLs following the same pattern will have the three features set to 100.

2) Lexical: We next derive five lexical features from the biggest cluster formed previously. Intuitively, we expect the *Path_length* feature to be short, a crucial requirement for a URL shortening service. Short paths are usually achieved by using numerical characters and uppercase letters to enlarge the available character space. Such a character composition is captured by the *Digit_ratio_** and *Uppercase_ratio_** features.

Common website URLs have paths in the form of a sequence of semantic strings that convey the main context of a webpage. Besides being intuitive to the visitor, semantic URLs play an important role in search engine indexing. In contrast, short URLs contain algorithmically generated pseudorandom sequences of characters. The *Vowel_ratio_** and *Bigram_score_ratio_** features measure the randomness of a path string. For the latter, we compute the frequencies of all possible character bigrams in an English language text corpus. Then, for each path string, we extract the substrings of length

two with a sliding window method. Using the precalculated frequencies, we compute the mean of substring frequencies and log transform it to get the bigram score. Finally, we compute the Hamming distance between each consecutive URL pair in the cluster, represented by *Hamming_distance_ratio_**.

3) Redirection-Based: This set of features is again computed for URLs in the biggest cluster only, using the redirection chains built previously. We expect URL shortening services to exhibit a high level of redirections to external domains, so we represent the number of redirecting URLs in the cluster using *Nb_redirecting_urls* and *Redirecting_url_ratio* features. To distinguish dedicated USS in classification (those that redirect to external domains of a single brand or company), we retain the number of unique target domains as a feature (*Redirecting_url_ratio*). Note that this feature set should not be affected by individual short URLs taken down due to abuse complaints or blocklist appearances, as the URLs are not sampled from blocklists, but rather the Wayback Machine.

B. Classifiers

We train two machine learning models (logistic regression and random forest) to determine the one performing the best in identifying the domain names of URL shorteners.

1) Logistic Regression: It transforms a linear combination of input features into a probability value indicating the likelihood of the output class. The algorithm assumes that data samples are independent of each other, the dataset is linearly separable, and there exists a linear relationship between the input features and the output class. Logistic regression considers that each class is equally represented in the dataset. If we train the model on an imbalanced dataset, the model tends to be biased towards the majority class (negative class in our dataset). Since our data is inherently imbalanced, we employ weighted logistic regression which assigns greater weight to the minority class, thereby imposing higher penalty on the model for errors made on minority class samples.

2) Random Forest: This is an ensemble machine learning model that fits multiple decision trees on the dataset and uses majority voting to predict the output class. Each tree is trained on random subsets of data and features preventing overfitting and improving the performance. This algorithm does not need feature normalization or standardization, as it ranks them based on their importance to provide insights into the feature selection. However, it is harder to interpret than logistic regression. Each decision tree in the random forest is built on a bag of uniformly distributed random samples. Therefore, each decision tree will be biased towards the majority class in the bag. To make the random forest less sensitive to class imbalance, we use a balanced random forest classifier.

C. Evaluation

We define the positive class as generic USS domains and the negative class as non-USS domains. Precision, recall, and the F_1 score are used as evaluation metrics. Precision quantifies the positive predictions that actually belong to the positive class. Recall refers to the proportion of correctly predicted positive instances over all positive instances. F_1 score is the harmonic mean of precision and recall.

TABLE II
BINARY CLASSIFICATION PERFORMANCE OF MACHINE LEARNING ALGORITHMS USING 5-FOLD CROSS-VALIDATION. THE EVALUATION METRICS TAKE INTO ACCOUNT THE CLASS IMBALANCE OF THE DATASET.

Method	**Precision**	**Recall**	F_1 **score**
Logistic regression	95.6%	96.7%	96.0%
Random forest	98.4%	98.0%	98.2%

$$\text{Precision} = \frac{TP}{TP+FP} \qquad \text{Recall} = \frac{TP}{TP+FN}$$

$$F_1 score = \frac{2TP}{2TP+FP+FN}$$

To evaluate the generalization of our models, we apply the repeated stratified k-fold cross-validation with 100 repeats of 5-folds. The basic k-fold validation randomly divides the data into chunks (folds) for which one might have negligible or no sample from the minority class. The stratified sampling, in turn, keeps the class distribution of the whole dataset when splitting into subsets. Table II shows the results for the two models. We observe that random forest performs better (with respect to the above metrics) than the logistic regression, suggesting that our data is better separated non-linearly. We therefore apply the trained random forest classifier on unlabeled data in Section V. We nevertheless use the logistic regression model to show the feature importance below as it provides interpretable coefficients.

D. Feature Analysis

We present the features along with their coefficients in the logistic regression model in Figure 2. The variation is computed for 10 repetitions of 5-fold cross-validation runs. We normalize the input feature values to prevent the domination of high value scales. Moreover, we use the Lasso regularization to remove redundant features and prevent overfitting. Lasso reduces multicollinearity by promoting sparsity in coefficients.

Larger absolute coefficient values indicate a stronger relationship between the feature and the predicted class. We can observe that *Nb_target_domains* is the strongest indicator of a URL shortening service. The analysis supports the observation that URLs from a generic shortening service tend to redirect to a wide variety of domains. The *Nb_urls_clustered* feature has the second highest coefficient. It reflects our assumption that short URL syntax similarity is a distinctive feature to identify USSs. Among lexical features, *Vowel_ratio_std* has the highest importance in classification implying that the distribution of vowels in short URLs exhibit great variability. The coefficients of six features are shrunk to zero, indicating high correlation with other features. This suggests that they do not provide additional discriminative power. The use of Lasso leads to a more compact and interpretable model by retaining only the most relevant features.

The sign of the coefficient shows the correlation between the feature and the positive class ("is a URL shortener domain").

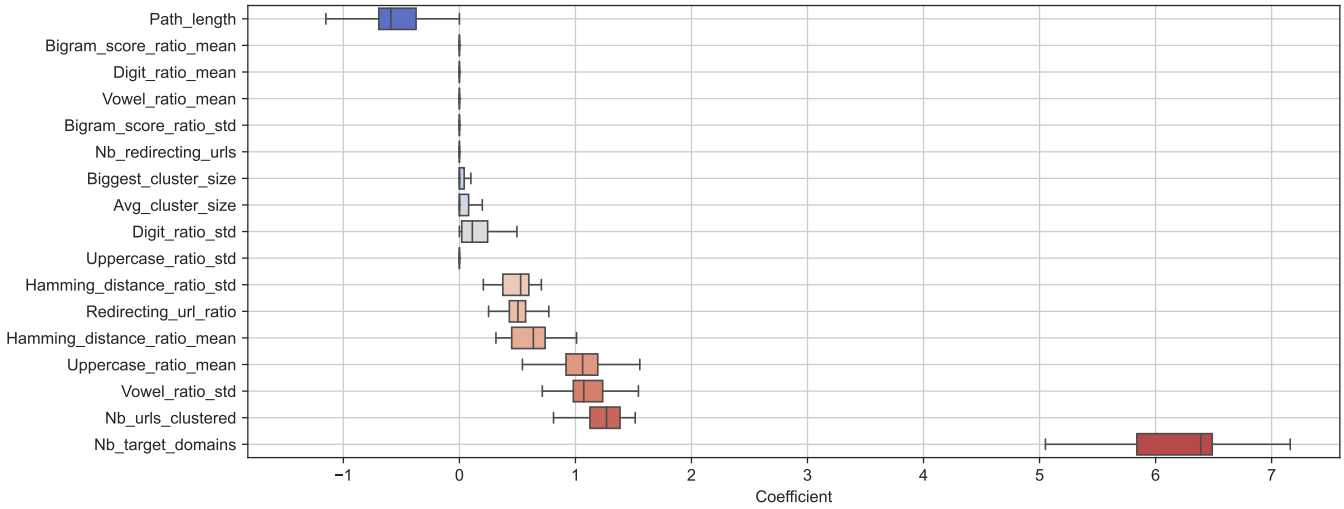

Fig. 2. Feature coefficients in the logistic regression model. The Lasso regularization sets six feature coefficients to zero, excluding them from the model. The high number of external redirection targets inside the same cluster is the strongest indicator of the domain being a USS.

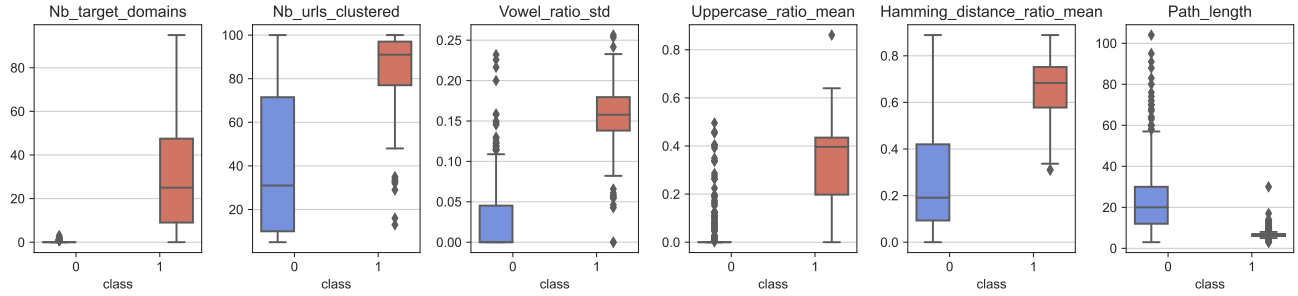

Fig. 3. Distributions of features with the highest coefficients. The number of target domains and the path length distinguish the two classes the most.

For example, the *Path_length* feature has negative coefficient meaning the longer the path length is, the higher the probability of the domain belonging to the negative class ("not a URL shortener"). Intuitively, service providers aim to keep the URL as short as possible.

Furthermore, Figure 3 presents the distributions of the six features with the highest coefficients. The number of target domains (*Nb_target_domains*) showcases the largest difference between the two classes: non-USS domains rarely redirect externally, but short links within the same cluster have up to 95 distinct target domains. The median number of URLs forming clusters (*Nb_urls_clustered*) is 91 for shortening services, but 31 for non-USS domains. We can also see that URLs from non-USS domains rarely contain uppercase letters (*Uppercase_ratio_mean*). The path length (*Path_length*) is another apparent feature, being only 7 characters long on average for short URLs but up to 104 for other links.

V. URL Shortening Services in the Wild

In this section, we apply the previously built classifier to phishing URLs reported by various blocklists. We want to evaluate the real-world detection rate of the classifier as it achieved high performance metrics on the labeled dataset. Having complemented our ground truth with more USS domains, we assess the prevalence and popularity of various URL shortening services in phishing as well as the rates at which they mitigate abuse.

A. Candidate Domains

We first collect 1.5 M URLs from three blocklist feeds provided to us by the Anti-Phishing Working Group (APWG) [42], OpenPhish [43], and Phishtank [44]. Our data collection spans the period between December 2023 and December 2024. We parse all the URLs and extract 312 k uniquely registered domains. Then, we perform DNS **A** record lookups and keep the domains returning the **NOERROR** response code. We next check them against the ground truth dataset and exclude 115 already labeled USS domains. This narrows down the list to 100 k active domains for further analysis.

B. Application on the phishing dataset

Having compiled the list of candidate domains, we apply the methodology described in Section III. For each domain in the list, we start by collecting historical URLs from the Wayback Machine. Setting the previously defined threshold of 100 URLs captured in the span of the last five years,

we keep 11.8 k domains meeting this requirement. We then perform pattern clustering of the retrieved URLs and keep those forming at least one pattern cluster narrowing down our candidate list to 6,468 domains. Finally, we perform the redirection measurement on URLs in the biggest clusters and remove those for which all URLs return some error response, keeping 2,937 domains.

We then extract the features listed in Table I and apply the random forest classifier previously trained on the labeled dataset. The detection threshold is set to 0.8 to obtain optimal trade-off between true positive and false positive rates. The model classifies 195 domains as being URL shortening services. We then manually verify all of them and reveal that 177 are true positives (correctly classified) and 18 are false positives, showing that the classifier performed with precision of 91%. Here we do not provide recall as it requires labeling of the whole dataset or having prior knowledge of the class distribution in phishing blocklists. We focus on the model precision.

Inspecting the domains and hosted websites manually, we observe that FPs are limited to certain types of services. First, four domains serve as dedicated USS having the same lexical and cluster-based features as a typical URL shortening service. For example, **`docdro.id`** hosts the dedicated shortening service for the file sharing website **`https://www.docdroid.net`**. Second, using historical crawls on the Internet Archive [35], we identified nine domains that used to serve as USS but they are deactivated possibly for one of the reasons outlined in Section II-D (link expiration or takedown). For instance, the **`shortly.at`** and **`tny.im`** services were active until late 2024 and early 2025, respectively. The remaining domains are webpage hosting services (**`dzr.io`**, **`page.co`**), the link-in-bio service (**`linkpages.pro`**), the ad network (**`thedailyelkcitian.com`**) and the link analyzer (**`xip.li`**).

These findings suggest that our approach is effective in identifying URL shortening services in the wild. However, as commonly observed in real-world detection tasks, its precision is lower compared to the evaluation on labeled data because of the presence of ambiguous services, which may not have been well represented in our training dataset. In particular, two-third of the FPs correspond to dedicated or discontinued shortening services, which share similar characteristics with generic URL shorteners.

C. USS Misuse in Phishing

Having enhanced our ground truth USS list with newly detected domains, we analyze the misuse of shortening services in phishing attacks. Overall, there are 15.9 k short blocklisted URLs from 293 URL shortening services, making up 1% of the phishing dataset described in Section V-A. Table III shows the ten most abused URL shortener domains, accounting for 70% of the blocklisted short links observed during the measurement period. Eight out of ten entries belong to the top 7 k domains ranked by the Tranco list (as of March 2025), therefore being generally popular services.

TABLE III
TEN MOST WIDELY USED URL SHORTENER DOMAINS OBSERVED IN THE PHISHING DATASET OF 1.5 MILLION URLS. THE FIFTH ENTRY (**`RB.GY`**) IS ALSO THE HIGHEST RANKED BY THE TRANCO POPULARITY LIST.

#	Domain	Tranco rank	URLs
1.	tinyurl.com	255	1,498 (12.9%)
2.	urlz.fr	41,213	1,391 (9.2%)
3.	cutt.ly	2,244	1,369 (9.0%)
4.	bit.ly	6,730	1,145 (7.6%)
5.	rb.gy	71	1,115 (7.4%)
6.	rebrand.ly	3,475	824 (5.5%)
7.	shorturl.at	6,319	809 (5.4%)
8.	is.gd	2,781	787 (5.2%)
9.	q-r.to	241,099	643 (4.2%)
10.	t.ly	3,654	583 (3.9%)

Figure 4 further presents the number of short phishing URLs per each of the top ten domains between January and December 2024. Each bar represents the weekly number of blocklisted URLs along with the brand targeted by the phishing attack. The **`tinyurl.com`** service is regularly exploited in various phishing campaigns making up almost 13% of all the abused short URLs. The second most abused service, namely **`urlz.fr`**, is consistently used in the phishing campaign targeting the French bank Crédit Agricole. Phishers seem to prefer the domain registered under the **`.fr`** country-code TLD to make the attacks look more credible. The United States Postal Service (USPS) and DHL are also commonly impersonated by phishers using several USS simultaneously. Some URL shortening services are not uniformly abused over time. For example, the use of **`rebrand.ly`** increased substantially in October 2024 with the phishing campaign targeting Facebook. The **`q-r.to`** service was only actively used between July and November 2024.

We observe that Bitly accounts for 8% of all the short URLs—a significant decrease since 2016, when APWG reported [45] half of the short phishing URLs using **`bit.ly`**. This decline could be a result of their adopted threat detection system—they keep a database of abused URLs, updated from three sources: trusted partners (e.g., Google Web Risk [46]), an internal threat detection system, and a publicly available abuse report form [47]. However, it was shown that one can evade their internal detection system by first setting up a benign destination page to later modify the content [9], as the system crawls and classifies the destination link at short URL creation time.

D. Uptimes of Malicious Short URLs

We measure the uptime of a malicious URL as the period between being blocklisted and the abuse mitigation (i.e., deactivation of a short link) by the URL shortening service. We do not take into account the mitigation of the target (landing) phishing pages pointed to by short URLs. Therefore, the registered domain takedown, account suspension by a hosting provider, and malicious content removal from destination websites are all outside the scope of this analysis.

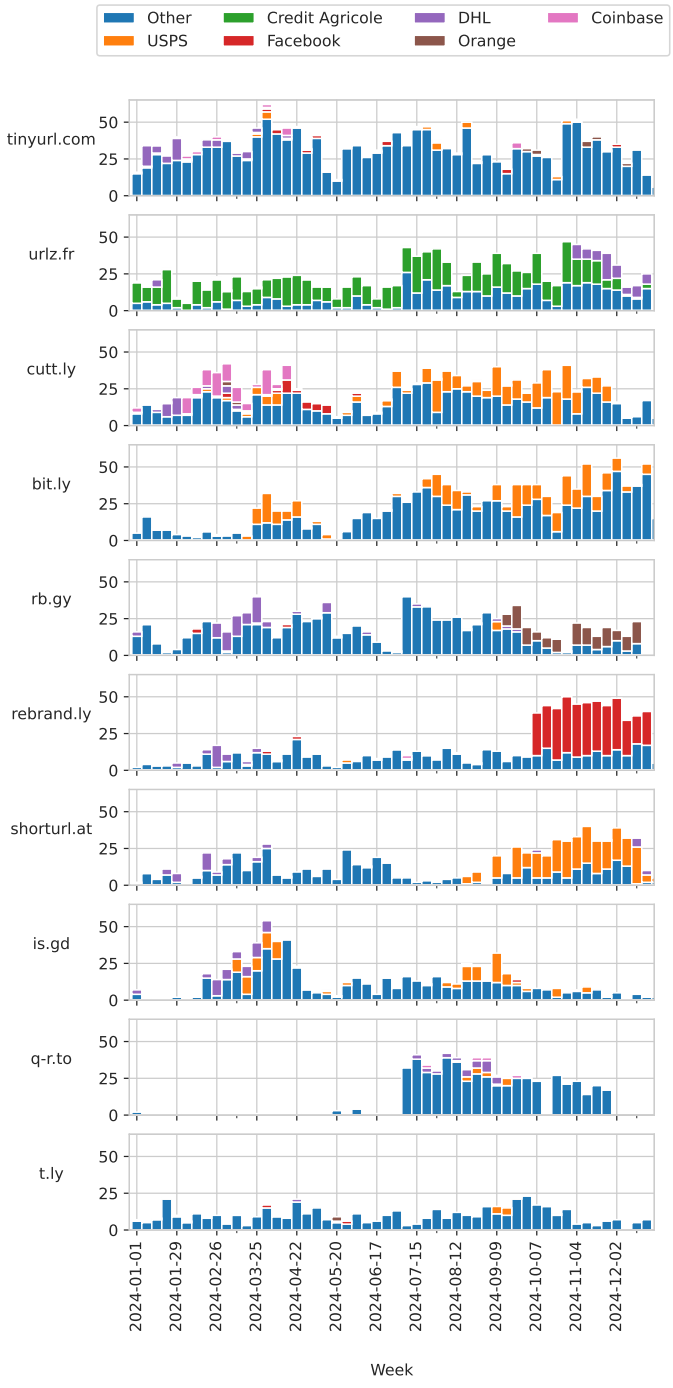

Fig. 4. Distribution of short links in phishing attacks under ten most popular USS—each bar represents the number blocklisted URLs during the corresponding week. Different colors distinguish the targeted brands.

We actively collect HTTP status codes and the HTML content of the blocklisted URLs as well as the DNS records and WHOIS data of the underlying registered domain. We note, however, that these measurements are resource- and storage-intensive, limiting our capability to conduct them for all the short URLs over an extended period of time. Our measurements, therefore, cover a 15% sample of the short URLs blocklisted between January 2024 and January 2025. For each short link, we perform the first measurement the moment we fetch it from the phishing blocklist. Depending on the provider, the feed is updated every five to 60 minutes, meaning the maximum delay between being blocklisted and measured does not exceed one hour. We run the subsequent measurements at 5/15/30 minutes, then hourly for 6 hours, and every 12h for one month. We stop the measurements afterwards, even if the short URL is still active and redirecting. Our rationale is that one month is sufficient time for various intermediaries to collect actionable evidence and act on the abuse reports.

TABLE IV
NUMBER OF SHORT URLS UNDER TOP 10 MOST MISUSED PROVIDERS AND THE RATIO OF DEACTIVATION. THE `Q-R.TO` USS SHOWS THE HIGHEST ABUSE MITIGATION RATE.

Rank	Domain	Total measured	Deactivated
1.	tinyurl.com	187	156 (83.4%)
2.	bit.ly	244	208 (85.2%)
3.	urlz.fr	93	42 (45.2%)
4.	cutt.ly	82	65 (79.2%)
5.	rb.gy	93	46 (49.4%)
6.	rebrand.ly	196	142 (72.4%)
7.	is.gd	24	18 (75.0%)
8.	shorturl.at	86	24 (24.0%)
9.	q-r.to	309	296 (95.7%)
10.	t.ly	31	26 (83.8%)

We observe that deactivated short URLs exhibit different behaviors depending on a particular service. For example, some return the **`404`** HTTP status code while others serve a page with the warning about the potential security risk at the destination. We therefore focus on the top 10 misused URL shortening services and manually verify how they mitigate malicious URLs. In particular, we download deactivation web pages and compute their **`ssdeep`** hashes, commonly used to match *almost* identical files (i.e., a slight change in the source file does not alter the hash significantly). Next, we compute the hashes of the content measurements and compare them to the deactivation page. When the page content is more than 70% identical to the mitigation page, we consider the short URL to be deactivated.

Table IV shows the total number of unique URLs covered by the uptime measurement and the number of URLs deactivated by the URL shortening service operator within one month. The **`q-r.to`** service has the highest deactivation rate of over 95%, while six more services deactivate over 70% of the blocklisted short URLs. Focusing on the remaining active short URLs, the manual analysis reveals that the destination pages were mitigated at the DNS or hosting levels, therefore, being unreachable even with a functioning short link.

Figure 5 shows the uptimes of malicious short URLs from the ten most abused USS. Overall, the median mitigation time is within 48h after blocklisting, with the exception of **`is.gd`**. Although having the highest deactivation rate, **`q-r.to`** is one of the slowest services to address abuse. Interestingly, the service owned by the same parent company, **`bit.ly`**, is one

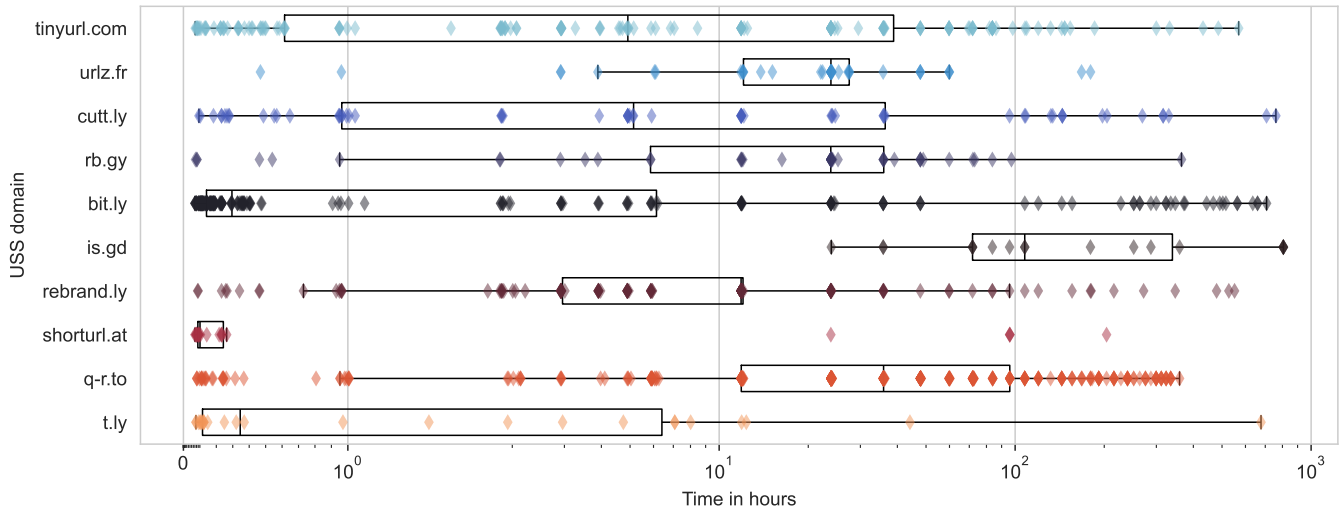

Fig. 5. Uptimes of abused short URLs. Boxplots represent the time elapsed between the short URL being blocklisted and deactivated. The data in the range from 0 to 1 is in the linear scale, while the rest is logarithmic. The left and right ends of the box mark 25th and 75th percentiles, and the line inside is the 50th percentile. We only have few samples from **`t.ly`**, **`is.gd`**, and **`shorturl.at`** so the results are not fully representative of the service uptimes.

of the fastest to react. Half of the short URLs from **`bit.ly`** are deactivated within 30 minutes after being blocklisted. We observe that **`cutt.ly`** is also fast to mitigate. It has a safe redirecting system in place that checks the destination URL against blocklists before redirecting [48]. Similarly, **`rebrand.ly`** uses a third-party solution [49] to detect abuse. The free version of the same service, **`rb.gy`** has a slightly slower response to abuse.

In summary, although most URL shortening services eventually deactivate malicious links, their mitigation responsiveness varies considerably. Even if services deactivate most malicious URLs, delays in mitigation can still leave users exposed to phishing attacks for extended periods. The observed variations in mitigation times suggest that non-technical factors such as the differences in abuse reporting mechanisms, service policies, or internal resource allocation, may significantly influence the effectiveness.

VI. Related Work

In this section, we review the related work in the field. We first summarize the research on distinguishing maliciously registered domains from compromised ones. We then focus on a specific class of misused services, namely URL shorteners.

A. Classification of Malicious Domains

Researchers proposed several machine learning models to determine whether malicious web contents are hosted under compromised websites or maliciously registered domain names [14], [50], noting how this attribution affects mitigation actions. Yet, they excluded special-use domain names, such as URL shortening services, subdomain providers, and dynamic DNS services from their analysis.

Silva et al. [51] built a classifier to distinguish domains behind malicious URLs as either public or private. They considered a domain to be public if its subdomains or path suffixes are not managed by the domain owner, including subdomain providers (e.g., **`000webhostapp.com`**) and URL shorteners (e.g., **`bitly.com`**) in the ground truth dataset. However, their features were only derived from domains and subdomains, resulting in several URL shortener domains (e.g., **`bit.ly`**, **`rebrand.ly`**) being misclassified as private domains. Our work fills this gap by classifying URL shortener domains as a separate class.

B. Analysis and Classification of Short URLs

Numerous studies mostly focused on Bitly [52], [13], [11], [10], [12], as it provided a convenient API to get various analytics data, including click rates, targets, and referrers.

Antoniades et al. [52] collected short URLs by crawling Twitter and brute-forcing hashed paths under **`bit.ly`** and **`ow.ly`**. The authors analyzed the obtained entries across various dimensions, including lifetimes, popularity, and influence on web performance. They found that half of the short URLs were ephemeral, being active during a couple of days only. The same year, Chhabra et al. [13] studied phishing attacks using **`bit.ly`** links on Twitter. They took the 2010 Phishtank dataset and found 6,474 phishing short URLs. Similarly to previous work, they analyzed the referrers, publishers, as well as the geographical and temporal spread of clickers using Bitly and Twitter APIs. They observed that phishers mainly target social media and e-commerce platforms. Le Page et al. [12] leveraged the Phishtank and X-force blocklists to identify malicious **`bit.ly`** links. They found that 1.45% of the URLs blocklisted in 2016 and 2017 were short. They compared the lifetimes and the click distribution of short URLs used in phishing and malware, showing that phishing links receive higher click-through rate but have shorter lifespan (50% remain active less than 80 days).

Leveraging machine learning approaches, Wang et al. [11] trained classifiers using the click traffic dataset to identify `bit.ly` spam short URLs. The developed random tree model reached the accuracy of 90.81% and the F_1 score of 0.91. Gupta et al. [10] proposed a different machine learning approach to classify `bit.ly` short URLs into malicious and benign. They used analytics features provided by the service and attained the accuracy of 86.41%.

Back in 2013, Maggi et al. [9] crowdsourced short URLs via a browser add-on. They acquired 25 M short URLs during two years of measurements and identified 622 USS, including branded services. The short URLs were then checked against four blocklists: Spamhaus, Wepawet, Google Safe Browsing, and Phishtank. The authors identified 44,936 malicious short URLs, out of which 3,806 were involved in phishing attacks. They showed that `bit.ly` was the most abused service followed by `tinyurl.com` and `ow.ly`. This approach, however, raises privacy concerns, as browser add-ons access all the web page content. More recently, Zhang et al. [5] compiled a list of 88 dedicated USS (DUSS) of well reputed domain names. They reported that a quarter of the services had link shortening APIs vulnerable to misdirection attack. The problem was partially fixed after the notification campaign.

Existing research has put a lot of effort into analyzing the use of short URLs, whether for benign or malicious activities. Yet, no work systematically collected generic URL shortening services at scale—without relying on crowdsourcing and making them available for the research community to use. Our paper fills this research gap by building a ground truth dataset of confirmed USS and training a machine learning classifier to identify similar services in the wild.

VII. Limitations of the Results

The results of this study have certain limitations. We manually curated the ground truth dataset and visited each website of the candidate URL shortening service. This process is tedious and time-consuming, making it challenging to perform at scale. We nevertheless prioritized the quality of the labeled dataset, even if it results in a smaller number of entries.

We then collected historical URLs from the Wayback Machine, which may not provide enough data for further analysis. Despite some domains having less than 100 or no links at all, we do not lower this threshold to collect a representative sample of the domain URLs. We otherwise risk to detect fewer pattern clusters and infer false URL shortening services.

Finally, we automatically visit all the collected links with the aim of detecting the redirection behavior. However, we observed that certain URL shortening services show destination preview pages and require a button click to visit the link. In fewer cases, users may also face a CAPTCHA challenge. Unfortunately, our automated approach cannot detect these types of redirections.

VIII. Ethical Considerations

Internet measurements require careful consideration of potential ethical concerns. In our work, we refer to the established industry practices that guide researchers in performing such experiments.

Wayback Machine provides URLs that may also contain user information parts. When retrieving historical URLs, we first parse each URL to remove the parts containing user credentials such as username and password before saving the list on our machines.

To refrain from overloading web servers during both data collection and redirection measurement, upon receiving the `429 Too Many Requests` or `529 Service is overloaded` response codes from a server, we respect the value in the `Retry-After` header. If the header is not provided in the response, we use a custom wait time of 60 seconds before retrying again.

IX. Conclusions

In this paper, we proposed and evaluated a robust method to automatically identify generic URL shortening services (USS) using supervised machine learning, significantly outperforming existing manual or crowdsourced approaches. Our random forest classifier, trained on a carefully curated ground truth dataset, achieved the 98.4% precision and maintained a strong performance (91% precision) when applied to an extensive real-world dataset of phishing URLs. We identified 177 previously unknown URL shortening services actively abused in phishing attacks, underscoring both the prevalence of these services in malicious activities and the necessity for precise detection mechanisms.

Moreover, our longitudinal measurement study revealed notable discrepancies among URL shortening providers in terms of responsiveness to abuse. While the median time to mitigate malicious short URLs across major providers is around 48 hours, some services, despite high deactivation rates, exhibited significant delays in mitigation, leaving end-users exposed to phishing attacks for prolonged periods. These findings indicate that provider-specific policies, internal resources, and abuse handling mechanisms likely play crucial roles in shaping mitigation effectiveness.

Overall, our results emphasize the need for URL shortening services to adopt clearer and more responsive abuse mitigation strategies, alongside increased transparency regarding their internal processes. For the broader security community, including registrars, hosting providers, and security teams, our automated classification approach offers a powerful tool for identifying abused shortening services at scale, enabling more targeted and timely interventions. Finally, we plan to make the list of inferred URL shortening services available to vetted researchers.

Acknowledgments

This work has been funded by KOR Labs and partially supported by the European Union Digital Europe Programme and the European Cybersecurity Competence Centre under Grant Agreement No. 101128042 (project ThreatChase). We thank the Anti-Phishing Working Group, PhishTank, OpenPhish for providing data access for this analysis.

REFERENCES

[1] T. Berners-Lee, L. M. Masinter, and M. P. McCahill, "Uniform Resource Locators (URL)," RFC 1738, 1994.

[2] A. Vaha-Sipila and E. Wilde, "URI Scheme for Global System for Mobile Communications (GSM) Short Message Service (SMS)," RFC 5724, 2010.

[3] X, "Counting characters when composing Tweets," 2025, https://docs.x.com/resources/fundamentals/counting-characters.

[4] TinyURL, "About TinyURL," 2025, https://tinyurl.com/app/features/about-us.

[5] Z. Zhang, L. Zhang, Z. Zhang, G. Hong, Y. Zhang, and M. Yang, "Misdirection of Trust: Demystifying the Abuse of Dedicated URL Shortening Service," in *Network and Distributed System Security Symposium*, 2024.

[6] Bitly, "A Day with Bitly," https://bitly.com/pages/resources/infographics/a-day-with-bitly, 2025.

[7] TinyURL, "Campaign Monitoring & Analytics," https://tinyurl.com/app/features/link-analytics, 2025.

[8] Bitly, "Empower your audience to start connecting with your brand—anywhere, anytime," https://bitly.com/pages/solutions/digital-marketing, 2025.

[9] F. Maggi, A. Frossi, S. Zanero, G. Stringhini, B. Stone-Gross, C. Kruegel, and G. Vigna, "Two Years of Short URLs Internet Measurement: Security Threats and Countermeasures," in *International Conference on World Wide Web*, 2013.

[10] N. Gupta, A. Aggarwal, and P. Kumaraguru, "Bit.Ly/Malicious: Deep Dive into Short URL Based e-Crime Detection," in *APWG Symposium on Electronic Crime Research (eCrime)*, 2014.

[11] D. Wang, S. B. Navathe, L. Liu, D. Irani, A. Tamersoy, and C. Pu, "Click Traffic Analysis of Short URL Spam on Twitter," in *9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*. IEEE, 2013, pp. 250–259.

[12] S. Le Page, G.-V. Jourdan, G. V. Bochmann, J. Flood, and I.-V. Onut, "Using URL Shorteners to Compare Phishing and Malware Attacks," in *2018 APWG eCrime*. IEEE, 2018, pp. 1–13.

[13] S. Chhabra, A. Aggarwal, F. Benevenuto, and P. Kumaraguru, "Phi.sh/$ocial: the Phishing Landscape Through Short URLs," in *Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference*, 2011, pp. 92–101.

[14] S. Maroofi, M. Korczyński, C. Hesselman, B. Ampeau, and A. Duda, "COMAR: Classification of Compromised versus Maliciously Registered Domains," in *IEEE EuroS&P*, 2020.

[15] A. Neumann, J. Barnickel, and U. Meyer, "Security and Privacy Implications of URL Shortening Services," in *Proceedings of the Workshop on Web 2.0 Security and Privacy*, 2010.

[16] J. Ji, "Awesome Url Shortener," 2024.

[17] P. D. Hello, "URL Shorteners," https://github.com/PeterDaveHello/url-shorteners, 2024.

[18] Polr, "Polr is a quick, modern, and open-source link shortener," 2025, https://polrproject.org.

[19] YOURLS, "The de facto standard self-hosted URL shortener," 2025, https://yourls.org.

[20] Shlink, "The definitive self-hosted URL shortener," 2025, https://shlink.io.

[21] A. Hancock, "What IP addresses does Bitly use?" 2023, https://support.bitly.com/hc/en-us/articles/230648007-What-IP-addresses-does-Bitly-use.

[22] M. Korczyński, M. Wullink, S. Tajalizadehkhoob, G. C. M. Moura, A. Noroozian, D. Bagley, and C. Hesselman, "Cybercrime After the Sunrise: A Statistical Analysis of DNS Abuse in New GTLDs," in *ASIACCS*. ACM, 2018, p. 609–623.

[23] S. Tajalizadehkhoob, T. van Goethem, M. Korczyński, A. Noroozian, R. Böhme, T. Moore, W. Joosen, and M. van Eeten, "Herding Vulnerable Cats: A Statistical Approach to Disentangle Joint Responsibility for Web Security in Shared Hosting," in *CCS*. ACM, 2017, pp. 553–567.

[24] N. Nikiforakis, F. Maggi, G. Stringhini, M. Z. Rafique, W. Joosen, C. Kruegel, F. Piessens, G. Vigna, and S. Zanero, "Stranger Danger: Exploring the Ecosystem of Ad-based URL Shortening Services," in *Proc. of the Web Conference,*, 2014, pp. 51–62.

[25] A. Hancock, "Will the links I create on Bitly ever expire?" 2024, https://support.bitly.com/hc/en-us/articles/360002288272-Will-the-links-I-create-on-Bitly-ever-expire.

[26] TinyURL, "How long will my TinyURL be available?" 2020, https://helpdesk.tinyurl.com/faqs/do-tinyurls-expire.

[27] Dario, "How Does Link Expiration Work?" 2024, https://support.rebrandly.com/hc/en-us/articles/360002915158-How-Does-Link-Expiration-Work.

[28] Jowi, "How to Create Links That Expire After a Number of Clicks," 2024, https://blog.short.io/how-to-create-links-that-expire-after-a-number-of-clicks/?utm_source=chatgpt.com.

[29] Dub.co, "Report Abuse," 2023, https://dub.co/legal/abuse.

[30] Buffer, "Report Start Page or content published via Buffer," 2023, https://support.buffer.com/article/662-report-start-page-or-content-published-via-buffer.

[31] tny.im, "tny.im has shut down," https://tny.im/, 2025.

[32] D. Czartoryski, "What happens if Sniply goes out of business? What happens to my links?" 2024, https://support.snip.ly/hc/en-us/articles/20001137039124.

[33] K. Reitz, "Requests: HTTP for Humans," 2025, https://requests.readthedocs.io/en/latest/.

[34] Microsoft, "Playwright enables reliable end-to-end testing for modern web apps." 2025, https://playwright.dev.

[35] Internet Archive, "Wayback Machine," 2025, https://web.archive.org.

[36] ——, "Wayback Machine APIs," 2013, https://archive.org/help/wayback_api.php.

[37] KOR Labs, "URLShortener," https://github.com/korlabsio/urlshortener, 2024.

[38] A. K., "What IP addresses does Short.io use?" 2024, https://help.short.io/en/articles/4065824-what-ip-addresses-does-short-io-use.

[39] G. Fiore, "Rebrandly IP Addresses / Firewall configuration," 2024, https://support.rebrandly.com/hc/en-us/articles/4765706843933-Rebrandly-IP-Addresses-Firewall-configuration.

[40] Piano, "Socialflow - Setting Up a Custom Short Domain," 2024, https://docs.piano.io/socialflow-setting-up-a-custom-short-domain/.

[41] V. L. Pochat, T. V. Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen, "Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation," in *NDSS*, 2019.

[42] Anti-Phishing Working Group, Inc, "APWG — Unifying The Global Response To Cybercrime," 2025, https://apwg.org/.

[43] OpenPhish, "OpenPhish - Phishing Intelligence," 2025, https://openphish.com/.

[44] Cisco Talos Intelligence Group, "PhishTank — Join the fight against phishing," 2025, https://phishtank.org/.

[45] G. Aaron and R. Rasmussen, "Global Phishing Survey: Trends and Domain Name Use in 2016," https://docs.apwg.org/reports/APWG_Global_Phishing_Report_2015-2016.pdf, 2017.

[46] Google Cloud, "Bitly: Ensuring real-time link safety with Web Risk to protect people," 2025, https://cloud.google.com/customers/bitly.

[47] Bitly, "Reporting Abuse of Bitly Links," https://bitly.com/pages/trust/report-abuse, 2025.

[48] Cuttly, "Is Cuttly safe?" https://cutt.ly/resources/support/get-started/is-cuttly-safe, 2025.

[49] Rebrandly, "Spam Link Auto Detection," https://support.rebrandly.com/hc/en-us/articles/115006483647-Spam-Link-Auto-Detection, 2024.

[50] J. Bayer, B. C. Benjamin, S. Maroofi, T. Wabeke, C. Hesselman, A. Duda, and M. Korczyński, "Operational Domain Name Classification: From Automatic Ground Truth Generation to Adaptation to Missing Values," in *Passive and Active Measurement*, 2023.

[51] R. D. Silva, M. Nabeel, C. Elvitigala, I. Khalil, T. Yu, and C. Keppitiyagama, "Compromised or Attacker-Owned: A Large Scale Classification and Study of Hosting Domains of Malicious URLs," in *USENIX Security Symposium*, 2021.

[52] D. Antoniades, I. Polakis, G. Kontaxis, E. Athanasopoulos, S. Ioannidis, E. P. Markatos, and T. Karagiannis, "we. b: The Web of Short URLs," in *Proc. of the Web Conference*, 2011, pp. 715–724.