

USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

BlockMeNot: Automatic Selection of Domain and URL Blocking Granularity to Minimize Collateral Damage and Evasion

Daud Ahmed, *IMDEA Software Institute/Universidad Politécnica de Madrid;* Srdjan Matic, *Universidad Carlos III de Madrid;* Platon Kotzias and Emiliano Carlesi, *BforeAI;* Juan Caballero, *IMDEA Software Institute*

https://www.usenix.org/conference/usenixsecurity26/presentation/ahmed

This paper is included in the Proceedings of the 35th USENIX Security Symposium.

August 12–14, 2026 • Baltimore, MD, USA

ISBN 978-1-939133-58-8

Open access to the Proceedings of the 35th USENIX Security Symposium is sponsored by



جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology

ARTIFACT EVALUATED
USENIX
AVAILABLE

BlockMeNot: Automatic Selection of Domain and URL Blocking Granularity to Minimize Collateral Damage and Evasion

Daud Ahmed
IMDEA Software Institute
Universidad Politécnica de Madrid
daud.ahmed@imdea.org

Srdjan Matic
Universidad Carlos III de Madrid
srdjan.matic@uc3m.es

Platon Kotzias
BforeAI
platon@bfore.ai

Emiliano Carlesi
BforeAI
emiliano@bfore.ai

Juan Caballero
IMDEA Software Institute
juan.caballero@imdea.org

Abstract

Domain and URL blocking is a fundamental mechanism for mitigating malicious, illegal, and inappropriate online content. Yet, selecting the right blocking granularity remains a largely unexplored problem. Overblocking can cause severe collateral damage if the content to be blocked uses the infrastructure of benign services, which may be unintentionally disrupted. Underblocking instead enables trivial evasion. This paper addresses this gap by proposing a novel approach that, given a domain or URL to be blocked, automatically selects the most appropriate blocking granularity (URL, subdomain, or apex) to minimize collateral damage and evasion opportunities. At the core of our approach are two novel machine learning classifiers to identify URL-leasing and subdomain-leasing apexes. The classifiers are trained and evaluated using a manually labeled dataset of 10,843 apexes, achieving F1 scores over 0.9. We implement our approach into BlockMeNot, a tool for identifying potential collateral damage prior to enforcement. We evaluate BlockMeNot on 225,355 entries (i.e., domains and URLs) collected from six blocklists and two threat exchanges. Although only 2.6% of apexes belong to leasers, they host 34.7% of the entries, and up to 71.7% in phishing-focused blocklists, underscoring the importance of fine-grained blocking. BlockMeNot identifies 1,976 leasing apexes, 59.3% of which were not in our ground truth, demonstrating the limitations of static lists and the need for our classifiers. Only 37.2% of blocklist entries are listed at the optimal granularity, 1.3% produce collateral damage, and 61.6% enable easy evasion. Overall, our work provides the first automated solution for domain and URL blocking granularity selection and offers practical guidance for block requesters, executors, and blocklist maintainers.

1 Introduction

Domain and URL blocking is a critical defense against malicious, illegal, and inappropriate resources on the Internet. A wealth of research has explored different aspects of this problem such as the blocklists [29,43,63,67,93] and threat exchanges [84] through which malicious resources to be blocked are distributed, and the takedown of malicious domains [5,50]. However, a fundamental question regarding domain and URL blocking has not been thoroughly investigated, namely, how to select the most appropriate blocking granularity.

The blocking granularity is fundamental for minimizing evasion and collateral damage. Underblocking enables trivial evasion. For example, an attacker can easily circumvent a blocked URL on a domain it has registered (i.e., an *apex*) by moving the malicious content to another URL or subdomain under the same apex. Overblocking causes instead painful collateral damage. For example, attackers can abuse *URL-leasing* services, which host content generated by their users on URLs under their apex (e.g., URL shorteners, web form creators), or *subdomain-leasing* services, which allow users to claim a subdomain under their apex (e.g., dynamic DNS providers, blog platforms). Blocking the abused service's apex not only blocks the malicious content but also benign content from other users of the service. ICANN frequently discusses the challenge of collateral damage in DNS blocking recommendations [35, 36, 66]. But, manual checks do not scale, and no automated approach exists to identify collateral damage. As such, highly impactful overblocking cases are frequently reported [12, 27, 41, 42, 55, 73]. For example, in February 2011, the US Government seized 10 domains involved in child pornography [26]. One of them belonged to a popular dynamic DNS provider, resulting in its 84K subdomains being blocked, even if the vast majority were unrelated to the block's intended scope [27].

In this work, we propose to automatically select the blocking granularity for each domain or URL to be blocked, so that collateral damage and ease of evasion are minimized on a case-by-case basis. In detail, our approach recommends a blocking granularity based on the type of apex in the input domain or URL. For apexes belonging to URL-leasing services, *URL blocking* is recommended so that only selected resources under the apex are blocked, preventing collateral damage. For apexes belonging to subdomain-leasing (but not URL-leasing)

services, *subdomain blocking* is recommended, which blocks all resources under a specific subdomain without affecting other subdomains. Finally, for apexes that do not lease URLs or subdomains, *apex blocking* is recommended, which blocks all subdomains and resources under the apex.

We have implemented our approach into BlockMeNot, a tool that given a domain or URL to be blocked, automatically recommends a blocking granularity (i.e., URL blocking, subdomain blocking, apex blocking). BlockMeNot can be used to determine potential collateral damage prior to URL and domain blocking by block requesters (e.g., victims, brand protection companies), block executors (e.g., DNS providers, firewall administrators, AV vendors), and blocklist managers. The blocking granularity an executor can apply depends on its location, e.g., URL blocking is only possible at the endhost, but not through DNS and rarely on network proxies. Still, the block executor should understand that it is about to block a URL-leasing service, and that subdomain or apex blocking may produce painful collateral damage. In fact, sometimes it may be best not to block and instead report the malicious resources to the URL-leasing or subdomain-leasing service so that they are removed, making blocking unnecessary.

At the core of BlockMeNot are two novel machine learning (ML) classifiers that allow identifying previously unknown URL-leasing and subdomain-leasing apexes. To train and evaluate our classifiers we produce a manually labeled ground truth (GT) of 10,843 apexes. The classifiers achieve 0.919 and 0.905 F1 scores for subdomain-leasing and URL-leasing apex identification, respectively. Previous work by De Silva et al. [79] proposed a classifier to jointly identify URL-leasing and subdomain-leasing apexes, which they called public apexes. However, accurate blocking requires differentiating both types of leasers because blocking a URL-leasing apex (e.g., `github.com`) at the subdomain level (e.g., `www.github.com`) would produce huge collateral damage (e.g., blocking all GitHub repositories). Also related is a recent classifier by Odgerel et al. [61] to identify URL shorteners. However, many other types of URL-leasing services exist beyond shorteners (e.g., web hosting, QR code generators).

We evaluate BlockMeNot on 225,355 entries (i.e., domains and URLs), belonging to 75,557 apexes, collected from 6 blocklists and two threat exchanges over one month. BlockMeNot recommends apex blocking for 65.2% of the entries and 97.4% of the apexes, subdomain blocking for 22.9% entries and 1.9% apexes, and URL blocking for 11.8% entries and 0.7% apexes. While only 2.6% apexes lease subdomains or URLs, those apexes host 34.7% of the analyzed malicious URLs and domains. For phishing blocklists, the percentage is significantly larger, reaching 71.7% for PhishTank, illustrating how web hosting is extremely attractive to phishers. The fact that 11.8% entries belong to URL-leasing services, illustrates the importance of distinguishing URL-leasing from subdomain-leasing apexes, so that the former can be applied URL blocking, minimizing the collateral damage.

Over one month, BlockMeNot identified 1,976 leasing apexes: 1,433 subdomain-leasing, 517 URL-leasing, and 26 leasing both. Of those 1,171 (59.3%) were previously unknown, i.e., are not in our GT. Even after three weeks, BlockMeNot still discovers new leasers every day, highlighting the limitations of static lists and justifying the need for our ML classifiers. Only 37.2% of entries are listed at the optimal granularity. We identify 1.3% entries that produce collateral damage, most of them in a blocklist aggregator (MalwareWorld) and a threat exchange (OTX). Using a dataset from a popular DNS provider, we show that such errors can impact tens of thousands of users over four days. Surprisingly, 61.6% of entries enable easy evasion because blocking the exact listed URLs offers little protection. But, generalizing the block from single URLs to coarser granularity risks introducing collateral damage, justifying the need for BlockMeNot.

This paper provides the following contributions:

- We propose a novel approach to automatically select URL and domain blocking granularity to minimize collateral damage and evasion.
- We design two ML classifiers to identify URL-leasing and subdomain-leasing apexes.
- We produce a dataset with 10,843 apexes manually labeled as subdomain-leasing, URL-leasing, or private.
- We implement BlockMeNot and apply the tool to identify previously unknown URL-leasing and subdomain-leasing apexes among 225,355 entries collected from six blocklists and two threat exchanges.
- We release our artifacts [11].

2 Blocking Granularity

An *apex* is the highest level of a domain name that can be registered and resolved, without any subdomains. For example, `www.example.com` and `blog.example.com` are two subdomains under the `example.com` apex. Extracting the apex of a domain requires using the ICANN part of the Public Suffix List (PSL) [58], which enumerates all standard gTLDs (e.g., com, net), country-code ccTLDs (e.g., uk, jp), and multi-level ccTLDs (e.g., co.uk, gov.in). The apex of a domain is the level above the highest level in the PSL's ICANN part. For example, since `co.uk` is part of the PSL's ICANN part, then the apex for `www.amazon.co.uk` is `amazon.co.uk`. A *domain* can be an apex or a subdomain of an apex.

Subdomain leasing. Some services allow their users to lease subdomains under an apex that belongs to the service. For example, Google's Blogger service allows users to choose the name of their blog (e.g., *mycoolblog*) and to host it under a subdomain of Blogger's `blogspot.com` apex (e.g., `mycoolblog.blogspot.com`). We call these *subdomain-leasing* apexes.

The PSL contains a private part where domain owners can add their subdomain-leasing apexes to let browsers know that each subdomain should be considered a separate site. However, the PSL's private part misses many subdomain-leasing apexes such as popular dynamic DNS providers [81].

URL leasing. Many services host content uploaded by different users under the same domain. We call these *URL-leasing* apexes because each resource URL (e.g., `apex.com/resourceA` and `apex.com/resourceB`) may point to content created by a different user of the service. For example, GitHub offers content created by different users under `github.com`. Other examples include URL shorteners, social media platforms, form builders, public forums, and file sharing services.

Some apexes may lease both subdomains and URLs. For example, Amazon AWS buckets can be accessed through a dedicated subdomain (e.g., `https://BUCKET.s3.sa-east-1.amazonaws.com`) and through a dedicated URL (e.g., `https://s3.sa-east-1.amazonaws.com/BUCKET/`).

Blocking granularity. A malicious resource can be blocked at different granularity levels: by blocking the resource's URL, its subdomain, its apex, or its IP addresses. This work focuses on URL, subdomain, and apex blocking. While the problem of selecting the right blocking granularity also exists with IP addresses, the blocking granularities are very different (e.g., IP subnets, ASNs) and require a separate approach. Three important factors when blocking a malicious domain or URL are: (1) the *collateral damage* (i.e., whether the blocking may affect other benign resources), (2) the *ease of evasion* (i.e., how easily the attacker can bypass the blocking), and (3) the blocking *location* (e.g., DNS, firewall, end-host).

Apex blocking. The coarser granularity is blocking all subdomains and resources underneath an apex. Apex blocking is well-suited for attacker-registered and compromised apexes since it cannot be evaded by creating new subdomains and resources underneath the apex. However, it is problematic if the apex leases subdomains or URLs because other benign subdomains and resources hosted on the same apex will also be blocked. Apex blocking can be performed on the DNS infrastructure, the end host, and on the network by leveraging the TLS Server Name Indication (SNI) header, if available (e.g., TLS 1.3 sessions may encrypt the SNI header using the Encrypted Client Hello extension [89]).

Subdomain blocking. The middle granularity is blocking all resources under a subdomain. However, if the attacker controls the apex, it can evade subdomain blocking by using different subdomains under the apex. Furthermore, if the domain leases URLs then benign content may be blocked, introducing collateral damage. Subdomain blocking is appropriate when the apex leases subdomains, but does not lease URLs. Similar to apex blocking, subdomain blocking can be performed at the end host or in the network.

URL blocking. The finest granularity is to block specific malicious resources, i.e., a subset of the URLs underneath a domain. Such blocking has minimal collateral damage. However, evasion is trivial if the attacker can create arbitrary subdomains under the apex or arbitrary resources underneath the blocked domain. URL blocking is appropriate with apexes that lease URLs (e.g., `github.com`) to avoid blocking an entire service due to an abusing user. Different granularities exist for URL blocking. For example, given `github.com/badUser/badRepo/badFile.exe`, one could block that exact URL, or `github.com/badUser/badRepo/*`, or `github.com/badUser/*`. Our approach will recommend URL blocking because the `github.com` apex leases URLs, but it will not specify which specific resources to block. URL blocking cannot be performed on the DNS infrastructure, and rarely on the network given the prevalent use of HTTPS, but can be performed on the end hosts prior to encryption.

3 BlockMeNot: Overview & Early Stages

Figure 1 illustrates the architecture of BlockMeNot, which given a malicious URL or domain, returns the most appropriate type of blocking, namely apex blocking, subdomain blocking, or URL blocking. BlockMeNot comprises nine modules: *Apex Extractor*, *Cache Resolver*, *Reputation*, *Enumerator*, *Liveness Checker*, *Private URL Shortener Fingerprinter*, *Subdomain-Leasing Classifier*, *URL-Leasing Classifier*, and *Blocking Selector*. The pipeline has a sequential flow with several early-exit conditions that trigger an immediate blocking recommendation.

First, the Apex Extractor obtains the apex from the input domain or the input URL's hostname. To extract the apex from a domain we use the *tldextract* library [45].

The Cache Resolver checks if the apex has been recently processed. If so, the cached blocking result is output without undergoing the rest of the pipeline. Caching minimizes the resources spent in re-classifying frequently reported apexes. The cache is initially populated with our ground truth (detailed in Section 4.1), new blocking decisions are added to the cache, and cached decisions are evicted after a configurable period.

The Reputation module checks if the apex is known to be malicious. For this, it queries the VT domain API to obtain the number of detections for the apex. If it has at least eight detections, then it is considered clearly malicious and apex blocking is recommended. We selected this conservative threshold since some leasers could have a few detections from security engines that have not yet identified them as benign services. If the apex has less than eight detections, is not known to VT, or no VT report is returned (e.g., due to query limits), the apex proceeds down the pipeline.

The Enumerator then produces a list of subdomains and URLs that belong to the apex. For this, it leverages external data sources such as VirusTotal [34], Common-

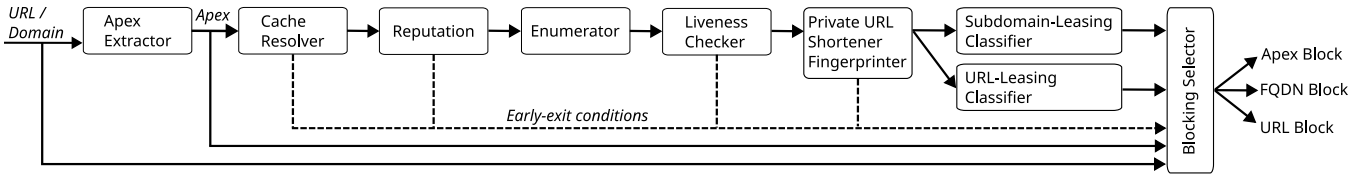


Figure 1: BlockMeNot architecture.

Crawl [22], WayBack Machine [94] and Certificate Transparency (CT) [32]. The Enumerator is detailed in Section 3.1.

Next, the Liveness Checker verifies that the apex is alive, i.e., the apex or any of its enumerated subdomains resolves and the apex is not parked. If the apex is considered dead, or less than a configurable number of live subdomains and URLs are identified, apex blocking is recommended. The Liveness Checker is detailed in Section 3.2.

Then, the Private URL Shortener Fingerprinter checks if the apex or its live subdomains run software used to build private URL shorteners, which only allow shortening URLs to authenticated users. The identification uses fingerprints detailed in Section 3.3. If a private shortener is identified, apex blocking is recommended, as attackers cannot easily abuse private services.

The core of our approach are the Subdomain-Leasing and URL-Leasing classifiers, which determine if the apex leases subdomains or URLs, respectively. We describe the design of the classifiers and the ground truth used to train and evaluate them in Section 4.

Finally, the classifier results are passed to the Blocking Selector, which outputs the recommended type of blocking. At a high level, if the apex leases URLs, then URL blocking is recommended. If it does not lease URLs, but leases subdomains, subdomain blocking is recommended. For apexes that do not lease URLs or subdomains, apex blocking is recommended. The blocking selector is detailed in Section 5.

3.1 Enumerator

The Enumerator produces a list of subdomains and URLs under a given apex, used by downstream modules to determine if an apex is alive and to obtain features for the classifiers.

Multiple open-source tools exist for enumerating subdomains [65, 71, 76] and URLs [46, 54, 72, 87]. These tools typically query external sources to identify subdomains and URLs (e.g., OTX [2], WayBack Archive [94], VirusTotal [34]). Some tools obtain their data from a single source, while others support a variety of free and commercial sources, among which the user selects the ones to use (providing credentials when needed). The tools are non-deterministic, i.e., different runs for the same apex may produce different enumerations since the data in the external sources may vary over time.

Among the available tools, we selected Subfinder [71] for subdomain enumeration and URLFinder [72] for URL enumeration because they support the largest number of sources, are written in the efficient Go language, and are frequently updated. Table 1 lists the sources we use. Most are publicly available, except for URLScan and VT that require a key.

Tool	Sources
Subfinder [71] only	Anubis, CertSpotter, Chaos, Chinaz, Crt.sh, Digitorus, DNSDumpster, DNSRepo, HackerTarget, HudsonRock, PugRecon, RapidDNS, Robtex, RSECloud, SiteDossier, THC, ThreatCrowd
URLFinder [72] only	URLScan
Both Subfinder and URLFinder	AlienVault OTX, CommonCrawl, VT API, WaybackArchive

Table 1: Data sources used by tools in Enumerator module.

Enumeration tools are slow due to the potentially large number of external sources queried, timeouts, and rate-limits. To minimize their use, we leverage datasets of URLs and domains collected from the VirusTotal (VT) URL feed [34] and Certificate Transparency (CT) [32], summarized in Table 2.

VT offers a commercial feed with the reports produced by analyzing URLs submitted by users. The industry authors collected two distinct 14-day datasets from this feed. VT1 has 37.8M domains extracted from all URLs submitted to VT between April 13 and April 26, 2025, but does not contain the URLs. VT2 contains 119.0M domains and 50.8M URLs corresponding to all subdomains and a maximum of 1,200 URLs for each apex in the URLs submitted between November 28 and December 12, 2025.

CT is an append-only public log of all certificates issued by contributing Certification Authorities (CAs). We collected two CT datasets, each containing all domains appearing in the Subject or Subject Alternative Name (SAN) fields of all certificates submitted to CT over a 14-day span. The dates for the CT datasets match those of the VT datasets. CT1 contains 200.5M domains from 56.7M apexes and CT2 has 254.5M domains from 74.2M apexes.

In detail, the Enumerator first extracts URLs for the given apex from the VT2 dataset. If more than 500 URLs are obtained, it outputs those URLs. Otherwise, it runs URLFinder on the apex and returns the union of the VT2 URLs and those discovered by URLFinder. Then, the Enumerator obtains subdomains for the apex from the URLFinder results and the

Dataset	Type	Start	End	Days	URLs	Domains	Apexes
Certificate Transparency (CT1)	Domain feed	2025-04-13	2025-04-26	14	-	200,576,489	56,794,894
Certificate Transparency (CT2)	Domain feed	2025-11-28	2025-12-12	14	-	254,502,759	74,238,050
VirusTotal (VT1) [34]	URL Feed	2025-04-13	2025-04-26	14	-	37,842,300	17,735,347
VirusTotal (VT2) [34]	URL Feed	2025-11-28	2025-12-12	14	50,815,319	119,083,473	18,130,810
AlienVault OTX (OTX) [2]	Threat Exchange	2025-12-13	2026-01-13	32	206,493	150,895	76,325
eCrime eXchange (eCX) [9]	Threat Exchange	2025-12-13	2026-01-13	32	79,579	-	19,164
PhishTank [82]	URL Blocklist	2025-12-13	2026-01-13	32	56,145	-	8,351
OpenPhish [64]	URL Blocklist	2025-12-13	2026-01-13	32	30,194	-	7,985
URLHaus [1]	URL Blocklist	2025-12-13	2026-01-13	32	7,056	-	956
UT Capitole [90]	Domain Blocklist	2025-12-13	2025-01-13	32	-	33,280	24,683
CertPL [69]	Domain Blocklist	2025-12-13	2026-01-13	32	-	18,699	14,702
MalwareWorld (MW) [68]	Domain Blocklist	2025-12-29	2026-01-13	16	-	1,609,095	1,191,331

Table 2: Dataset summary. The VT and CT datasets at the top are used by the Enumerator, described in Section 3.1. The threat exchange and blocklist datasets at the bottom are used to evaluate BlockMeNot in Section 6.

VT1, VT2, CT1, and CT2 datasets. If more than 500 subdomains are identified, it outputs those subdomains. Otherwise, it runs Subfinder on the apex, returning the union of the subdomains in the Subfinder and URLFinder results and the VT and CT datasets.

3.2 Liveness Checker

The Liveness Checker determines if an apex is alive. First, it resolves the apex and the enumerated subdomains. RFC 8020 [15] specifies that an NXDOMAIN response for a domain should indicate that no subddomains underneath the queried domain exist. However, we have found apexes that return NXDOMAIN but have subdomains that resolve, justifying the need to resolve all subdomains. If neither the apex nor any subdomain resolved (i.e., returned a valid IP address), the apex is considered dead.

If the apex resolved, heuristics by Zirngibl et al. [101] are used to determine whether it is parked. Parked apexes are considered dead. If the apex is not parked, or it did not resolve but some subdomains did, domain registration data is collected using RDAP and WHOIS. If the apex is expired or has *clientHold* or *serverHold* EPP status codes (indicating it has been taken down by a registry or registrar [5]), the apex is considered dead.

The moment an apex is considered dead, the Liveness Checker early-exits with an apex block recommendation. While dead domains may not need to be blocked, it is safer to block them in case some live subdomain was missed. If the apex is alive but less than 10 URLs and less than 5 subdomains are identified, then BlockMeNot early-exits with an apex blocking decision because not enough information is available to run the classifiers and the probability of the apex being a leaser is low. Otherwise, the Private URL Shortener Fingerprinter, is invoked.

Name	Type	GT Apexes
Premium URL Shortener [95]	SaaS	205
Yourls [97]	Open Source	156
TinyURL [86]	SaaS	136
Bitly [14]	SaaS	68
Targeter Pro [23]	Licensed	58
AdLinkFly [31]	Licensed	55
Linkvertise [48]	SaaS	42
Zat.ink [24]	Licensed	30
Kutt [83]	Open Source	22
Dub [28]	Open Source	10
Polr [99]	Open Source	10
Short.io [37]	SaaS	8
Rebrandly [75]	SaaS	6
Shlink [19]	Open Source	4
Short Menu [8]	SaaS	3
T2M [98]	SaaS	1

Table 3: URL shortening software detected by the Signature Matcher, ordered by apexes in GT.

3.3 Private URL Shortener Fingerprinter

URL shorteners are one of the most popular, and abused, URL-leasing services [61]. While building our GT, we identified many URL shorteners generated using URL shortening software. We also observed that not all URL shorteners were publicly available. Some were private, i.e., only allowed shortening URLs to authenticated users. Public shorteners lease URLs to third parties; thus they require URL blocking. Instead, private shorteners can only be used by authenticated users; thus they can be blocked at the apex level.

The Private URL Shortener Fingerprinter identifies URL shorteners built using 16 URL shortening software, summarized in Table 3, and determines if they are public or private. The goal is to identify private URL shorteners, which can be blocked at the apex level without going through the ML classifiers. This approach differs from that of Odgerel et al. [61],

whose classifier detects public URL shorteners. We instead construct fingerprints that automatically identify private URL shorteners built using popular URL-shortening software.

The supported software can be grouped into three classes: Software-as-a-service (SaaS), open-source, and licensed. Some popular URL shorteners (e.g., Bitly [14], TinyURL [86]) allow companies to easily deploy their own private shorteners through a Software-as-a-service (SaaS) offering. For example, the New York Times uses Bitly to provide a private shortener at `nyti.ms`. URL shortening open-source projects [28, 83, 97, 99] and licensed commercial software [23, 31, 95] allow deploying either public or private shorteners.

We rely on DNS and HTTP fingerprints to identify the 16 URL shortener software and determine if they are public or private. DNS fingerprints are used to identify private SaaS shorteners since the returned IP address or name server belong to popular URL shortening services (e.g., 67.199.248.0/24 for Bitly or ns*.tinyurl.com for TinyURL). HTTP fingerprints identify shorteners using open-source and licensed software. They access a URL under the apex and apply a signature on the HTTP response. For example, Yourls [97] installations serve error images with known file hashes, while Dub [28] responses include the "X-Powered-By: dub;" HTTP header.

When an open-source or licensed software is identified, a secondary HTTP-based fingerprint is used to determine if the shortener is configured to be public or private. Secondary fingerprints access an HTML resource and check for specific tags and values in the returned HTML, e.g., login forms. For example, *Premium URL Shortener* [95] private installations contain the string "Thanks for your interest but this website is currently used privately".

If a private shortener is identified, BlockMeNot early-exits with an apex blocking recommendation. When a public shortener is detected, BlockMeNot does not early-exit because the apex running a public shortener may be configured to provide shortened URLs on a different apex. For example, `shortenly.net` is a public shortener built using Targeter Pro [23], but the shortened URLs are on `shote.nl`. The fingerprinting module cannot distinguish that `shortenly.net` does not lease URLs, but `shote.nl` does. But, the URL-leasing classifier in Section 4 can, so apexes running public shorteners, and those not running shorteners, are passed to the classifiers.

4 Classifiers

This section details our subdomain-leasing and URL-leasing apex classifiers. Section 4.1 describes the ground truth used to train and evaluate the classifiers, Section 4.2 the classifiers' features, and Section 4.3 the model training and evaluation.

4.1 Ground Truth

To train and evaluate our classifiers we built a ground truth (GT) of labeled apexes. Our GT assigns three labels to apexes: subdomain-leasing, URL-leasing, or *private*, i.e., not leasing URLs nor subdomains. Apexes may be assigned both subdomain-leasing and URL-leasing labels.

Building the GT comprised two steps. First, we collected candidate leaser and private apexes from a variety of data sources. Then, we manually validated the candidates to ensure apexes were alive (i.e., resolved and not parked) and determine whether they leased subdomains, URLs, or both.

To obtain candidate leasers, we leveraged the private part of the PSL [58][1] for subdomain-leasing apexes; three public lists of URL shortener services [59, 60, 91]; and four datasets from previous works [39, 40, 78, 79]. We also built a crawler to obtain the public subdomain-leasing apexes from FreeDNS [6], a dynamic DNS provider that prior work identifies as having the largest number of apexes [81]. Finally, we queried the top 1M popular apexes according to the Tranco list[2] using the SpamHaus intelligence API [70] and selected apexes labeled by SpamHaus as dynamic DNS providers and URL shorteners. To obtain candidate private apexes, we used the top 140 apexes by number of subdomains in our CT dataset and sampled the Tranco 1M list at different heights (i.e., top 10K, 100K-200K, bottom 500K) for apexes not in other sources.

We manually validate all candidates to ensure a high quality GT. The manual validation first examined the apex's primary website to check its stated purpose. If the purpose was clear (e.g., URL shortening), we performed functional testing to verify that the leasing activity occurred on the apex being examined and not on alternative apexes (e.g., by creating test accounts). If the purpose was not clear (e.g., no website) we checked whether different resources (i.e., subdomains or URLs) under the apex hosted content that seemed generated by different entities. For this, we leveraged digital archives [10, 94] and search engines to obtain the content hosted on different resources and then manually determined if it seemed to come from different owners. A mix of malicious and benign content under an apex, as reported by VirusTotal [34], was also a signal for leasers.

At the end, our GT dataset comprises 10,843 apexes: 6,637 subdomain-leasing, 1,832 URL-leasing, and 2,394 private. Among these, 20 apexes lease both URLs and subdomains.

4.2 Feature Engineering

The classifier features capture characteristics of an apex related to its potential leasing behavior. Table 4 summarizes the 22 features used by our subdomain-leasing classifer and Table 5 the 18 features used by the URL-leasing classifier. One feature (apex *age*) is shared by both classifiers. Of the

[1] The PSL was downloaded on May 27, 2025.
[2] We use the Tranco list from April 12, 2025.

Feature	Category	Description	Prior	DPI
age	general	Domain age in days	[79]	3.44%
num_subdomains	general	Total enumerated subdomains	[18, 79]	3.36%
avg_subdomain_depth	general	Average subdomain depth	[18, 79]	5.11%
num_uniq_apex1	general	Unique apex+1 subdomains	✗	4.52%
frac_pop_keywords	lexical	Fraction of subdomains with popular keywords	[79]	24.68%
avg_sub_entropy	lexical	Average entropy of subdomain parts	[49, 79]	1.68%
apex_entropy	lexical	Entropy of apex	✗	2.10%
frac_three_char_prefixes	lexical	Ratio of unique 3-char prefixes to apex+1	✗	13.70%
med_sub_len	lexical	Median character length of subdomain lengths	✗	4.84%
mad_sub_len	lexical	Median absolute deviation of subdomain lengths	✗	0.23%
stdev_sub_len	lexical	Standard deviation of subdomain lengths	✗	4.57%
frac_live_subdomains	network	Fraction of live subdomains	✗	2.07%
frac_ips	network	Fraction of unique IPs to live subdomains	✗	1.60%
frac_port_80_or_443	network	Fraction of IPs with port 443 or 80 open	✗	5.14%
num_web_servers	network	Unique web servers identified	✗	0.44%
frac_ip_asns	network	Ratio of unique ASNs to unique IPs	✗	12.37%
frac_ip_subnets_24	network	Ratio of /24 subnets to resolved IPs	✗	1.60%
frac_cert_issuers	network	Ratio of distinct issuers to certificates	✗	1.86%
frac_cert_orgs	network	Ratio of distinct organizations to certificates	✗	0.82%
frac_wildcard_certs	network	Fraction of certificates with wildcards	✗	1.19%
frac_self_signed_certs	network	Fraction of certificates that are self-signed	✗	3.09%
frac_subdomains_with_certs	network	Fraction of subdomains with certificates	✗	0.34%

Table 4: Apex features used by subdomain-leasing classifier. Decrease in Performance Importance (DPI) measures feature importance in our best-performing Gradient Boosting model. A larger percentage indicates higher importance.

39 features, 5 are taken from previous domain classification works [18,49,79]. We exclude other features from those works because they do not capture properties of subdomain and URL leasers, or because they are specific to one dataset, e.g., features capturing total and malicious URL scans are specific to the commercial VT URL feed [79]. We selected potential new features using domain knowledge, evaluated them using feature selection metrics like Decrease in Performance Importance (DPI) and Leave-One-Feature-Out (LOFO), removing those that did not positively contribute to the classification.

Most features aggregate information collected from the subdomains and URLs returned by the Enumerator and Liveness Checker. Each time those modules run on an apex, different sets of subdomains and URLs may be returned. Furthermore, some features send network requests whose responses may vary over time. BlockMeNot caches blocking recommendations so that an apex's features are only recomputed when the apex is evicted from the cache and needs to be re-classified.

Because of space constraints, we do not describe individual features. Instead, we group them into three categories (general, lexical, and network) and discuss the rationale behind each category.

General features represent the apex age and volume, and structural properties of its subdomains and URLs. They capture that leasing services tend to live long, generate large numbers of subdomains and URLs, and assign subdomains at a specific domain label height. The *age* is the only feature used in both classifiers. It is computed using the WHOIS/RDAP creation date. If WHOIS/RDAP information is not available, we approximate the creation date using the earliest appearance of the apex in the following sources: Wayback Machine [94], CT logs [32], and domain information services [30, 57].

Lexical features capture that leased subdomains and URLs often include a user-selected part (e.g., username, repository), which manifests in linguistic properties such as component length, character distribution, and the use of specific keywords. For *num_pop_keywords*, since the list of popular keywords from [79] was not available, we use the top 100 subdomains in the collected CT logs.

Network features leverage information collected through DNS and HTTP requests. For efficiency, if an apex has more than 500 live subdomains or URLs, we compute its network features on a randomized sample of 500. These features capture (1) liveness since resources in leasers are created and removed frequently; (2) HTTP redirections characteristic of some leasers (e.g., URL shorteners, QR code generators); (3) the server infrastructure including the used IP addresses and autonomous systems (ASNs) and the prevalence of web servers; and (4) certificate management practices, which can reveal dynamic DNS providers where web servers on different subdomains have different owners.

4.3 Model Training & Evaluation

We partition the GT into training, validation, and testing subsets using stratified sampling with a 70/15/15 split. To miti-

Feature	Category	Description	Prior	DPI
age	general	Domain age in days	[79]	6.03%
num_urls	general	Total enumerated URLs	✗	2.00%
frac_url_subdomains	general	Fraction of URLs to unique subdomains	✗	14.95%
avg_num_path_slashes	lexical	Average number of slashes in the URL path	✗	1.30%
stdev_num_path_slashes	lexical	Standard deviation of slashes per path	✗	1.08%
mad_url_len	lexical	Median absolute deviation of URL lengths	✗	0.75%
stdev_url_len	lexical	Standard deviation of URL lengths	✗	2.99%
apex_len	lexical	Character length of the apex	✗	9.79%
path_diversity	lexical	Average Levenshtein distance of URL paths	✗	15.31%
frac_urls_with_ext	lexical	Fraction of URLs containing file extensions	✗	7.79%
frac_urls_with_params	lexical	Fraction of URLs with query parameters	✗	0.54%
frac_uniq_first_segments	lexical	Fraction of unique first path segments	✗	3.69%
frac_urls_with_hyphens	lexical	Fraction of URLs with hyphens in path	✗	1.97%
frac_resource_urls	lexical	Fraction of URLs with resources	✗	1.50%
avg_path_entropy	lexical	Average entropy of URL path	✗	11.16%
frac_url_apexes	network	Ratio of unique final apexes to total URLs	✗	17.04%
avg_redirect_chain_len	network	Average length of HTTP redirect chains	✗	2.03%
stdev_redirect_chain_len	network	Standard deviation of HTTP redirect chains	✗	0.08%

Table 5: Apex features used by URL-leasing classifier. Decrease in Performance Importance (DPI) measures feature importance in our best-performing Random Forest model. A larger percentage indicates higher importance.

Model	Acc.	Prec.	Rec.	F1	ROC	PR
RF	0.912	**0.909**	0.880	0.892	**0.973**	0.988
ET	0.909	0.904	0.878	0.889	0.972	0.988
GB	**0.918**	0.901	**0.909**	**0.905**	**0.973**	**0.989**
AB	0.881	0.864	0.850	0.856	0.946	0.975
LR	0.837	0.807	0.835	0.817	0.914	0.954
KNN	0.886	0.885	0.839	0.857	0.952	0.973
SVM	0.910	0.896	0.892	0.894	0.967	0.986

Table 6: Subdomain-leasing apex classification results on validation set using stratified 5-fold cross-validation.

Model	Acc.	Prec.	Rec.	F1	ROC	PR
RF	**0.962**	**0.970**	0.880	**0.918**	**0.981**	**0.944**
ET	0.959	0.967	0.875	0.913	0.979	0.936
GB	0.944	0.879	**0.921**	0.899	0.977	0.937
AB	0.935	0.874	0.874	0.874	0.947	0.874
LR	0.895	0.793	0.889	0.828	0.960	0.896
KNN	0.957	0.953	0.875	0.909	0.959	0.903
SVM	0.944	0.905	0.872	0.887	0.962	0.878

Table 7: URL-leasing apex classification results on validation set using stratified 5-fold cross-validation.

Classifier	Model	Prec.	Rec.	F1
Subdomain-leasing	GB	0.914	0.926	0.919
URL-leasing	RF	0.959	0.867	0.905

Table 8: Final classifier accuracy on held-out test set.

gate the GT class imbalance, we employ class weighting. For the subdomain-leasing classifier, the positive class are the GT entries labeled as subdomain leasers and the negative class the entries labeled as URL leasers and private. For the URL-leasing classifier, the positive class are the URL leasers and the negative class the subdomain-leasing and private apexes. For each classifier, we train seven models using the features described in Section 4.2: Random Forest (RF), Extra Trees (ET), Gradient Boosting (GB), AdaBoost (AB), Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Logistic Regression (LR). For the SVM, KNN, and LR models features are normalized with a standard scaler.

We use the validation dataset to select the best hyperparameters for each model via 5-fold cross-validated grid search. Table 6 summarizes the accuracy of the subdomain-leasing classifier with the different trained models. GB is the best-performing model in terms of accuracy, recall, F1 score, and precision-recall area under the curve (PR-AUC), while RF achieves best precision and ties for Receiver Operating Characteristic Area Under Curve (ROC-AUC). We select GB as the model used by BlockMeNot for subdomain-leasing identification. The GB model uses 1,500 estimators, a maximum depth of 4, square root feature selection, a 0.8 subsampling rate, and a learning rate of 0.01. Table 7 summarizes the accuracy of the URL-leasing classifier using the different models. In this case, the RF model tops all metrics, save for recall that is best with GB. We select RF as the model used by BlockMeNot for URL-leasing identification. The RF model uses 800 estimators, a maximum depth of 16, square-root feature selection, a minimum of 2 samples per split, and bootstrap sampling.

Table 8 summarizes the accuracy of the final classifiers on the held-out testing dataset using a 0.5 decision threshold. In production, this threshold can be adjusted to meet specific operational requirements, such as increasing precision at the

cost of recall. The GB subdomain-leasing classifier achieves precision of 0.914, recall of 0.928, and an F1 score of 0.919. The RF URL-leasing classifier achieves precision of 0.959, recall of 0.867, and an F1 score of 0.905.

Feature importance. We analyze feature importance in the selected models using Decrease in Performance Importance (DPI). This metric is model-agnostic and captures the percentage drop in accuracy when a feature is excluded. The DPI results are shown in the rightmost column of Tables 4 and 5. The larger the percentage, the more the feature matters. For the subdomain-leasing classifier, the use of popular keywords in subdomains, the subdomain prefixes, and the ratio of ASNs hosting servers are the most important features. For the URL-leasing classifier, the ratio of landing apexes after redirection over all URLs and the average URL edit distance matter most.

5 Blocking Selector

The Blocking Selector determines the final blocking recommendation. If the pipeline early-exits (i.e., the subdomain-leasing and URL-leasing classifiers are not run), then the blocking recommendation depends on the module that early-exited. If the input's apex was cached, then the cached blocking decision is output. If the enumerator did not extract enough subdomains or URLs, apex blocking is recommended because it is unlikely that the apex leases subdomains or URLs. If the liveness checker determined that the apex was dead, apex blocking is recommended, in case some unidentified subdomain exists and is alive. Finally, if the Private URL Shortener Fingerprinter identified that the input runs a branded URL shortener, apex blocking is recommended because branded shorteners are private and cannot be externally abused.

If the pipeline reaches the two ML classifiers, the blocking recommendation takes into account the output of both classifers. If the apex leases URLs, then URL blocking is recommended. If it does not lease URLs, but leases subdomains, subdomain blocking is recommended. For apexes that do not lease URLs or subdomains, apex blocking is recommended.

It is important to note BlockMeNot may recommend a finer blocking that the input type. For example, if the input is a domain, BlockMeNot may still recommend URL blocking if it identifies that the input's apex leases URLs. This is done to send a clear signal to the user about the collateral damage expected from blocking the provided input. For example, if the input to BlockMeNot is `github.com`, BlockMeNot will determine that this apex leases URLs and recommend URL blocking. This lets the user know that it should not be blocking the input apex. Rather, it should block the particular resource under GitHub that is considered malicious, or not to block at all if the specific malicious resource is unknown. This behavior is also privacy-sensitive. If a user wants to know if a specific URL containing a resource should be blocked, it can opt to only submit the apex to BlockMeNot. Based on the apex, BlockMeNot may still recommend subdomain or URL blocking, which the user can apply without revealing the specific malicious resource.

In addition to the blocking decision, BlockMeNot outputs additional information about the apex to justify the recommendation, e.g., whether the apex is alive or dead and whether it hosts a private URL shortener. Among this additional information, BlockMeNot includes the reputation of the apex using three lists: Chrome User Experience Report (CrUX) [33], Tranco [88], and Cloudflare Radar top 1M [21]. If the recommended decision is apex blocking, but the apex is in the top 100K of any of these lists, then a warning is output stating that even if the apex does not lease, blocking it may still generate significant collateral damage due to its popularity.

6 Real-World Evaluation

This section evaluates BlockMeNot. We present our evaluation datasets in Section 6.1, present BlockMeNot's blocking recommendations in Section 6.2, examine collateral damage and evasion cases in Section 6.3, and evaluate BlockMeNot's performance in Section 6.4.

6.1 Evaluation Datasets

To evaluate BlockMeNot, we collected URLs and domains published in six blocklists and two threat exchanges over one month starting on December 13, 2025. Our evaluation window starts after the model training window, which finished on December 12, ensuring no temporal bias. The evaluation datasets are summarized in Table 2.

Blocklists. Of the six blocklists. three provide URLs (PhishTank [82], OpenPhish [64], URLHaus [1]) and another three domains (CertPL [69], Malware World [68], UT Capitole [90]). The blocklists focus on malware (MalwareWorld, URLHaus, UT Capitole[3]), phishing (PhishTank, OpenPhish), or dangerous websites targeting a specific country (CertPL). MalwareWorld was not available December 13–28, so our collection started on December 29. MalwareWorld aggregates over a hundred other blocklists, thus its volume (1.6M domains) is an order of magnitude larger than the other sources. We exclude URLs containing IP addresses.

Threat exchanges. We also collect entries from two threat exchanges: the Anti-Phishing Working Group (APWG) eCrime Exchange (eCX) and the AlienVault Open Threat Exchange (OTX). In eCX, vetted contributors submit records of cybercrime-related activities accessible to all exchange members. We were provided research access to the *phish* endpoint that provides phishing URLs. In OTX, registered users submit *pulses* with indicators of compromise (IoCs) and

[3]While UT Capitole classifies URLs into many categories, we focus on the largest malware category.

Blocklist	All		Apex Blocking		Subdomain Blocking		URL Blocking	
	Entries	Apexes	Entries	Apexes	Entries	Apexes	Entries	Apexes
OTX*	81,688	14,953	68,219 (83.5%)	14,291 (95.6%)	10,725 (13.1%)	476 (3.2%)	2,744 (3.4%)	186 (1.2%)
eCX	79,579	19,164	58,926 (74.0%)	18,640 (97.3%)	19,722 (24.8%)	409 (2.1%)	931 (1.2%)	115 (0.6%)
PhishTank	56,145	8,351	15,886 (28.3%)	7,588 (90.9%)	18,244 (32.5%)	501 (6.0%)	22,015 (39.2%)	262 (3.1%)
UT Capitole	33,280	24,683	29,661 (89.1%)	24,298 (98.4%)	3,530 (10.6%)	362 (1.5%)	89 (0.3%)	23 (0.1%)
OpenPhish	30,194	7,985	18,426 (61.0%)	7,596 (95.1%)	10,544 (34.9%)	317 (4.0%)	1,224 (4.1%)	72 (0.9%)
CertPL	18,699	14,702	18,417 (98.5%)	14,594 (99.3%)	277 (1.5%)	105 (0.7%)	5 (0.0%)	3 (0.0%)
Mal.World*	11,497	7,575	10,300 (89.6%)	7,525 (99.3%)	1,179 (10.3%)	40 (0.5%)	18 (0.2%)	10 (0.1%)
URLHaus	7,056	956	3,659 (51.9%)	838 (87.7%)	1,067 (15.1%)	99 (10.4%)	2,330 (33.0%)	19 (2.0%)
Total	225,355	75,557	147,026 (65.2%)	73,581 (97.4%)	51,676 (22.9%)	1,433 (1.9%)	26,653 (11.8%)	543 (0.7%)

Table 9: Blocking recommendations output by BlockMeNot for blocklist and threat exchange entries published between Dec 13, 2025–Jan 13, 2026. MalwareWorld and OTX entries are a sample of all entries in the month.

metadata (e.g., date, title). We collect pulses using the OTX API following the approach by Allegretta et al. [4], i.e., first list the top 2,000 contributing users and then collect all pulses they submitted over the month. We extract URLs and domains from the collected pulses using iocsearcher [52].

DNS resolutions. A popular free DNS service provided us data on the number of DNS resolutions for all subdomains of 1,300 URL-leasing and subdomain-leasing apexes in our one-month dataset. For privacy reasons, the data does not include source IP addresses nor any other client identifiers. In detail, the dataset covers 672M DNS A and AAAA queries for 16.9M subdomains of the 1,300 apexes. The dataset covers a sample of 5% of all queries received by the DNS service over four days (Tuesday to Saturday) in January 2026, i.e., total queries for a domain over those four days may be 20 times larger. We use this dataset to rank apexes by popularity and to provide a lower bound on the impact of collateral damage.

6.2 Blocking Recommendations

We run BlockMeNot over all entries collected from six sources (CertPL, eCX, OpenPhish, PhishTank, URLHaus, UT Capitole). Due to the large volume of Malware World and OTX, we run BlockMeNot over the entries from 500 randomly selected apexes each day. Overall, we run BlockMeNot on 225,355 entries belonging to 75,557 apexes.

Table 9 summarizes the blocking granularity recommendations. Of the 225,355 entries, 147,026 (65.2%) received an apex blocking recommendation (i.e., belong to apexes that do not lease URLs or subdomains), 51,676 (22.9%) a subdomain blocking recommendation (i.e., belong to subdomain-leasing apexes), and 26,653 (11.8%) a URL blocking recommendation (i.e., belong to URL-leasing apexes). Thus, over one third (34.7%) of the malicious URLs and domains examined belong to URL-leasing or subdomain-leasing services, highlighting their prevalent abuse by attackers.

At the apex level, of the 75,557 apexes, 73,581 (97.4%) received an apex blocking recommendation, 1,433 (1.9%) a subdomain blocking recommendation, and 543 (0.7%) a URL blocking recommendation. While only 2.6% apexes lease subdomains or URLs, those apexes host 34.7% of the analyzed malicious URLs and domains.

The higher presence of entries in subdomain-leasing services (22.9%) compared to URL-leasing services (11.8%) may be due to those services offering higher flexibility to attackers. For example, dynamic DNS providers allow attackers to redirect subdomains to the their own servers where they can install any needed software and content. Still, 11.8% entries belong to URL-leasing services, illustrating the importance of distinguishing URL-leasing from subdomain-leasing apexes, so that the former can be applied URL blocking, avoiding the collateral damage from coarser blockings.

Three sources have an above-average ratio of entries in URL-leasing and subdomain-leasing services: PhishTank (71.7%), URLHaus (48.1%), and OpenPhish (39.0%). Both PhishTank and OpenPhish focus on phishing, indicating that phishers abuse leasing services significantly more than other attackers. The high prevalence of leasers in phishing sources means that if a user applied apex blocking to all their entries, the collateral damage would be devastating. In contrast, malware-focused blocklists exhibit a below-average ratio: CertPL (1.5%), Malware World (10.4%), and UT Capitole (10.9%). The difference may be partly due to phishers needing webpage hosting, offered by many services, while malware authors favor attacker-controlled hosting.

New leasers. Over the analyzed month, BlockMeNot identified 1,976 leasing apexes: 1,433 subdomain-leasing, 517 URL-leasing, and 26 that lease both. Of those 1,976 leasers, 805 (40.7%) were in our GT, while the remaining 1,171 (59.3%) were previously unknown. Out of the 1,171 new leasers, 258 (22%) have a Tranco ranking below 100,000. We manually examine half of those confirming that 87% of URL-leasers and 79% of the subdomain leasers are TPs, with the rest being potential FPs. FNs are harder to identify, although we observe one affecting a popular ad exchange. We discuss the impact of FPs and FNs in Section 7.

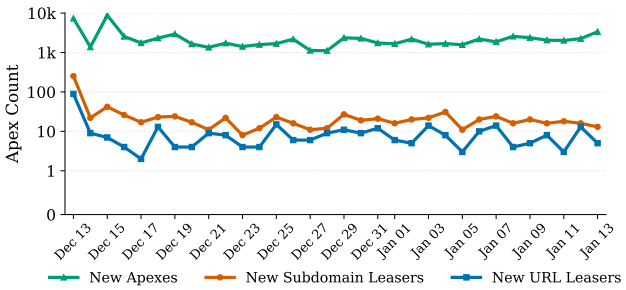


Figure 2: New apexes observed each day and the subset of those identified as new subdomain-leasing and URL-leasing apexes (logarithmic scale).

Figure 2 shows the new leasers identified each day. After the first few days, the number of new leasers identified daily stabilizes between 1 and 10. Even after three weeks, BlockMeNot still discovers new leasers every day. The continuous identification of new leasers justifies the need for our ML classifiers, since a static list could hardly keep pace.

Table 10 details the top 20 newly identified leasers ranked by number of sources listing entries for those apexes. The ranking is dominated by phishing entries since those appear in most sources. All 20 apexes belong to subdomain leasers. Of those, 13 are web hosting providers (e.g., `sviluppo.host`, `nazwa.pl`) with several offering free hosting (e.g., `netsons.org`, `iceiy.com`). Another 6 are website builders that allow quickly developing and deploying websites sometimes without using any code (e.g., `mocha.app`, `created.app`). The final one is a ccTLD (`eu.cc`) that was not in our GT because it was added to the PSL recently. If we instead focus on the malware-focused blocklists and the more general OTX, many other leasers beyond web hosting and builders appear such as QR code creation services that lease URLs (e.g., `q-r.to`, `scanned.page`), application monitoring software (`sentry.io`), network tunneling (`ply.gg`), and email marketing (e.g., `fdske.com`, `umeng.com`). All the above appear in at least two sources, but unfortunately it is not easy to understand how they are abused due to the lack of context in the listings.

6.3 Collateral Damage & Evasion

Next, we identify potential cases of collateral damage and evasion among the blocklist and threat exchange entries. For this, we classify each entry into three listing granularities: URL, subdomain, and apex. Table 11 shows the split of entries by their listing granularity and apex type according to BlockMeNot. The table captures 474,424 entries (instead of 225,355 in Table 9) because it includes all entries for MalwareWorld and OTX on the 75,557 apexes in our one-month evaluation. The diagonal in the table captures entries that are listed in what BlockMeNot believes is the best granularity. These correspond to 37.2% of all entries.

#	Apex	SL	UL	Entries	Queries
1	mdbgo.io	✓	✗	76 (7)	164
2	sviluppo.host	✓	✗	63 (7)	204
3	netsons.org	✓	✗	62 (7)	1,202
4	iceiy.com	✓	✗	53 (7)	209
5	mocha.app	✓	✗	48 (7)	163
6	appwrite.network	✓	✗	47 (7)	465
7	gitbook.io	✓	✗	552 (6)	12,601
8	ghost.io	✓	✗	160 (6)	73,820
9	eu.cc	✓	✗	141 (6)	2,793
10	created.app	✓	✗	67 (6)	247
11	wasmer.app	✓	✗	48 (6)	581
12	nazwa.pl	✓	✗	46 (6)	8,037
13	wuaze.com	✓	✗	30 (6)	1,871
14	edgeone.app	✓	✗	28 (6)	4,210
15	myclickfunnels.com	✓	✗	27 (6)	1,564
16	kesug.com	✓	✗	20 (6)	869
17	page.gd	✓	✗	18 (6)	258
18	1wp.site	✓	✗	18 (6)	102
19	ispot.cc	✓	✗	13 (6)	866
20	mybluehostin.me	✓	✗	13 (6)	900

Table 10: Top 20 newly identified subdomain-leasing (SL) and URL-leasing (UL) apexes ranked by number of sources listing them (in parenthesis), and the number of queries for the apex in the DNS dataset.

Apex Type	Listing Granularity		
	URL	Subdomain	Apex
URL-leasing	28,871 (6.1%)	4,632 (1.0%)	350 (0.1%)
Sub.-leasing	58,318 (12.3%)	68,233 (14.4%)	849 (0.2%)
Private	142,428 (30.0%)	91,654 (19.3%)	79,089 (16.7%)

Table 11: Split of entries by apex type and listing granularity. The diagonal (green) captures the most accurate listings. Below the diagonal (blue) are potential evasion cases. Above the diagonal (red) are potential collateral damage cases.

Below the diagonal are potential evasion cases. There are two main reasons for these. First, if the apex is controlled by the attacker (i.e., private) but the blocklist only lists specific subdomains, then the attacker can bypass the blocking by changing the malicious content to a separate subdomain. This case affects 19.3% of entries. For example, UT Capitole lists `vm05.transportrrj.com`, but the apex is attacker-controlled with 11 detections in VT, thus the attacker can move the malicious content to another subdomain. Second, when entries are URLs, if the apex is controlled by the attacker (30.0% of entries) or belongs to a subdomain-leasing service (12.3%), then blocking specific URLs allows for easy evasion. While URLs identify malicious resources at the finest granularity, they do not help block executors decide whether the blocking should happen at a coarser granularity. For example, given the following URL in URLHaus `https://hotelembuguacu.blob.core.windows.net`

`/data/<UUID>.pdf` (<UUID> is a macro that replaces a random identifier for safety), a user of the blocklist may decide to only block that resource. But, BlockMeNot determines that `windows.net` leases subdomains and thus recommends subdomain blocking. In detail, `blob.core.windows.net` is used by the Azure Blob Storage service [56]. All content under `hotelembuguacu.blob.core.windows.net` belongs to the same Azure user. Blocking only the listed resources enables evasion by changing the malicious content to another URL on the same subdomain. In fact, URLHaus lists two URLs matching the above pattern with different UUIDs.

Above the diagonal are collateral damage cases, which are rare (1.3% of entries) and only affect domain listings (i.e., URL blocklists and eCX are not affected), but can be highly impactful. One case is that a listed subdomain belongs to a URL-leasing service (1.2% of entries). For example, UT Capitole lists `file-eu-par-4.gofile.io`, which belongs to the GoFile cloud storage service. Blocking that subdomain, prevents file distribution from GoFile's Paris node. The DNS dataset shows 16.5K queries to gofile.io over four days, with 3.6K coming from France, highlighting the potential impact on users. In the other two cases, an apex is listed that belongs to a subdomain-leasing (0.2% entries) or URL-leasing (0.1% entries) service. Most of these cases are due to MalwareWorld (55.3%) and OTX (44.0%), with UT Capitole having 0.7% and CertPL none. MalwareWorld aggregates many blocklists and OTX receives contributions from any registered user. In both cases, highly popular apexes can creep into their listings (e.g., `github.com`, `google.com`). Aware of the issue, both services uses whitelists to alert their users of domains that should be ignored[4]. But, those whitelists are incomplete. For example, `mooo.com`, which belongs to the FreeDNS dynamic DNS provider [6], is not whitelisted by MalwareWorld, even if there was a famous incident in 2011 where the US Goverment blocked this apex, creating collateral damage to 84K subdomains [27]. Similarly, OTX's whitelist does not include `giize.com` and `3utilities.com`, both belonging to dynamic DNS providers. These three apexes are popular with `mooo.com` having 97.4K queries in the DNS dataset, followed by `giize.com` (12.3K), and `3utilities.com` (5.8K).

The results show that domain blocklists and threat exchanges can be occasionally impacted by collateral damage. Most of these cases affect blocklist aggregators (MalwareWorld) and threat exchanges (OTX), with very few cases affecting centrally-managed domain blocklists (CertPL, UT Capitole) likely because blocklist managers are vetting entries. In contrast, URL blocklists do not have collateral damage as they list entries at the finest granularity, but their entries can be trivially evaded and they do not offer assistance in how to make the blocking coarser.

[4] MalwareWorld seems to use Majestic [51] although this is not stated.

Module	Invocations	Avg. Time/Entry
Apex Extractor	7,269 (100.0%)	0.0086 ms
Cache Resolver	7,269 (100.0%)	0.3402 ms
Reputation	3,775 (51.9%)	14,520.8912 ms
Enumerator	2,439 (33.5%)	40,800.1071 ms
Liveness Checker	2,439 (33.5%)	148.9760 ms
Fingerprinting	759 (10.4%)	1,176.9158 ms
Classifiers	759 (10.4%)	2,285.5618 ms
Blocking Selector	7,269 (100.0%)	0.0065 ms

Table 12: Number of invocations and average time to process an entry for each module, measured over 7,269 entries.

6.4 Performance Evaluation

We evaluate BlockMeNot's performance using 7,269 entries collected from all sources on January 19, 2026. BlockMeNot runs on a server with an AMD Ryzen 9 9900X CPU and 64 GB of RAM. The initial cache is the one output by the one-month evaluation. Overall, BlockMeNot processed 166.4 entries per hour, completing the evaluation in 43.7 hours.

Table 12 details the number of invocations and the average time to process an entry for each module. All 7,269 entries are processed by the Apex Extractor and Cache Resolver, but only 3,775 apexes reach the Reputation module, i.e., the cache eliminates the need for further processing of 48.1% entries. The Reputation module requires 14.5 seconds per entry due to VT's rate limiting. It accounts for 35% of total execution time. Domains flagged as malicious by fewer than 8 security vendors proceed to enumeration. The Enumerator is the most expensive module, requiring 40.8 seconds per entry and accounting for 63% of the total execution time, due to querying multiple external sources with API rate limits. For the interested reader, Appendix A evaluates the contribution of the different enumeration sources, showing that some sources could be removed to improve efficiency. The Liveness Checker removes an additional 1,680 apexes that are dead or did not have enough subdomains and URLs to run the classifiers. The ML classifiers only process 759 apexes, 10.4% of the initial entries, requiring only 2.3 seconds to extract all the features and classify and apex. We discuss potential performance improvements in Section 7.

7 Discussion

This section discusses implications of our results, limitations, and avenues for future work.

Implications of our results. Our one-month evaluation shows that only 37.2% of blocklist and threat exchanges entries are listed at the granularity that minimizes collateral damage and evasion opportunities. Collateral damage is rare (1.3% entries), as it does not affect URL listings and rarely affects

well-managed centralized domain blocklists (i.e., CertPL, UT Capitole). However, even a single case can be devastating when it affects a popular service. Over 99% of the collateral damage instances identified affect a blocklist aggregator (MalwareWorld) and a threat exchange (OTX), reflecting the heterogeneous quality of the entries they aggregate. While those sources try to identify errors using whitelists, those are incomplete, opening the door to painful overblocking. BlockMeNot can assist the managers of such services, and their users, to identify problematic entries that should not be blocked. In contrast, 61.6% of the entries are at such fine granularity that evasion is simple. The bulk of these cases are URLs in subdomain-leasing and private apexes (42.3% entries). If the user simply blocks the listed URLs, evasion is trivial. Unfortunately, no assistance is provided for consumers of URL blocklists to determine the impact of blocking at coarser granularities. If the user decides to broaden the blocking (e.g., because it can only block on the DNS infrastructure or the network), it runs the risk of blocking a leaser, thus introducing collateral damage. BlockMeNot can assist blocklist consumers with identifying URLs in leasers.

Limitations. BlockMeNot has limited throughput due to the slow enumeration process. Despite using caching and early-exit conditions, the 166 entries/hour throughput prevented us from evaluating all entries in MalwareWorld and OTX. To address the Enumerator bottleneck, we could parallelize it, acquire commercial licenses with higher API query rates, and remove unnecessary sources. Appendix A details the contribution of each enumeration source, identifying those that could be removed. Another limitation is that when recommending URL blocking, our approach does not specify the set of resources to block, i.e., it determines that `github.com` should be blocked at the URL level, but not how to block specific users or repositories. Addressing this issue is an interesting problem, which may require analyzing the domain-specific URL structure.

Although both classifiers achieve F1 scores over 0.9, they are not perfect, suffering from false positives (FPs) and false negatives (FNs). Missing leasers (FNs) may cause BlockMeNot to recommend apex blocking, potentially introducing collateral damage. However, manual identification does not scale and also misses leasers, as demonstrated by frequently reported cases [12, 27, 41, 42, 55, 73]. Falsely identifying leasers (FPs) may enable evasion. However, over 61% of entries already allow easy evasion. Furthermore, given that only 2.6% apexes are classified as leasers, analysts may be able to manually validate those. Finally, in future work we will explore the temporal component: how our models degrade over time, when they should be retrained, and what is the optimal caching policy.

Compromised domains. Our approach suggests apex blocking for any apex believed to be controlled by the attacker, which includes both attacker-registered and compromised domains. For compromised domains, it is hard to determine the degree of control the attacker has achieved (e.g., injecting resources or creating arbitrary DNS entries). Thus, it is safer to entirely block the compromised apex temporarily, while the domain owner is notified and the domain is cleaned. In future work, we plan to explore how to integrate methods to detect compromised domains [53, 79].

Final blocking decision. In some occasions, the block executor may decide to override the blocking granularity suggested by BlockMeNot. For example, even if BlockMeNot identifies a leaser, the block executor may determine that the abuse of the service is so prevalent that it deserves coarser apex blocking. This happened in the takedown of the Nitol botnet with Microsoft being granted control over the `3322.org` dynamic DNS apex because it was widely abused [42].

8 Related Work

Abuse of public services. Previous works identify domains that lease resources to their users. De Silva et al. [79] proposed a classifier that uses data from the VT URL feed to jointly identify URL-leasing and subdomain-leasing apexes, which they called public apexes. Our work differs in differentiating subdomain-leasers from URL leasers, which is fundamental for accurate blocking granularity selection, and in using dataset-independent features. Also related is the URL shortener classifier by Odgerel et al. [61]. Our URL-leasing classifier identifies not only URL shorteners but a large variety of other URL-leasing services (e.g., web hosting, QR code generators), Also related are previous works that quantify the extent of malware abuse of benign Internet services [3, 96]. However, those works do not identify leasers.

Blocklists. Prior work has extensively studied the composition, coverage, and effectiveness of blocklists [29, 43, 47, 63, 67, 74, 80, 93, 100]. Some works analyze a variety of aspects of domain and URL blocklists such as the inclusion of benign domains, differences between free and commercial blocklists, the information provided about the listings, and coverage limitations [29, 43, 63, 67, 93]. Other works evaluate the effectiveness of blocklists against specific threats such as phishing [62, 63]. Our work complements those by analyzing a previously overlooked aspect, how to automatically select the domain and URL blocking granularity. Other works examined IP address blocklists [47, 74, 80, 100]. The problem of selecting the right blocking granularity also applies to IP addresses, but the blocking granularities are very different (e.g., IP subnets, ASNs), so our approach does not apply.

Threat intelligence sharing. A related line of work has investigated the sharing of indicators of compromise (IoCs) such as domains, IP addresses, and file hashes through public feeds [92], high-cost commercial feeds [16], and threat exchanges [17, 38]. Those works conclude that threat intel-

ligence sharing helps because different feeds (both free and commercial) have limited overlap, which points to each feed only covering a limited fraction of the entire threat ecosystem. In our work, we also observe the blocklist and threat exchanges used for evaluation to be complementary.

Malicious domain and URL detection. Also related are works proposing detectors to identify malicious domains [7, 13,25,44] and URLs [77,85] and those that evaluate them [20]. Such detectors can be used to populate the blocklists and threat exchanges that we use to evaluate our approach. Furthermore, some of the features used by our classifiers are inspired in malicious domain detection works [18, 49, 79].

9 Conclusion

This paper investigated the underexplored challenge of selecting the appropriate granularity for domain and URL blocking. We developed an automated approach that balances the risks of collateral damage and evasion by dynamically choosing between blocking at the URL, subdomain and apex level. By designing and validating machine-learning classifiers capable of identifying apexes that lease URLs and subdomains, we demonstrated that fine-grained decisions can be made accurately and at scale. Our large-scale evaluation across multiple blocklists and threat-intelligence sources highlights that a substantial fraction of current blocking practices either cause unnecessary disruption or remain trivially bypassable.

Our tool BlockMeNot, provides a practical solution for assessing blocking decisions prior to enforcement. Our results underscore the need for mechanisms to support block requesters, executors, and blocklist maintainers in making informed blocking granularity choices. Overall, this study establishes a foundation for more precise and accountable content-blocking practices on today's web.

Acknowledgments

We thank the maintainers of the services used for making their data public or providing the academic authors with research access. We thank Quad9 for sharing DNS resolution data that supported this research. Commercial services were only accessed by the industry authors and research access was only used by the academic authors. This work was partially funded by the Spanish Government MCIN/AEI/10.13039/501100011033/ through grants PID2022-142290OB-I00 (ESPADA), CEX2024-001471-M, and RED2024-154240-T (EMACS). Partial support was also provided by the Regional Government of Madrid through grant 2025-T1/COM-36349. The above grants are co-funded by European Union ESF, EIE, FEDER, and NextGeneration funds.

Ethical Considerations

The main stakeholders affected by our research are data subjects, data owners, and Internet users.

Data subjects. The only user data we collected are DNS queries to selected apexes provided by a large DNS provider. This dataset contains subdomains queried and the country from which queries originate, but it does not contain source IP addresses, any other user identifiers, or the IP addreses returned by the queries. Thus, we cannot link the queries to specific users. We only use the data to obtain query volumes per apexes and aggregated country distributions.

Data owners. We collect data from many services including blocklists, threat exchanges, VirusTotal, URLScan, and Certificate Transparency. Our data collection respected the terms of those services. We will not redistribute the collected data (e.g., URL entries, VT reports) but only aggregated data such as the features computed for the different apexes. We detail the specific time frame used from each source. Interested researchers can download or request access to the data from the services directly. Some of the data owners (e.g., blocklists, threat exchanges) are among the potential users that would benefit from our approach.

Internet users. Our research is justified in the need to identify the right domain and URL blocking granularity in order to minimize collateral damage and evasion. Minimizing those two factors offers benefits to the general Internet user population since they may be affected by blocking. While domain and URL blocking are fundamental to protect against malicious content in the Internet, they can also be used for unethical applications such as censorship. Our research does not enable new types of blocking or censorship, but it rather allows examining blocking impact so that block requesters and executers can make informed decisions. A potential negative impact of BlockMeNot would be errors where BlockMeNot recommends apex blocking for undetected leasers, potentially introducing collateral damage. Our tool design aims to minimize such errors. Furthermore, we argue that users are not worse with BlockMeNot than without it, as currently they lack strong mechanisms to identify leasers. This is a key reason behind previously reported overblocking cases that introduced significant collateral damage.

Open Science

To enable reproducibility of our research, we have released our artifacts [11]. The release includes the code of BlockMeNot, the blocking recommendations for the evaluated apexes, computed feature vectors, a subset of our ground truth dataset, and instructions for how reseachers can request access to the full ground truth under a non-commercial, non-redistribution, license. As detailed in the Ethical Considerations we cannot redistribute the data obtained from data owners such as the VT

URL feed, the eCX and OTX exchanges, and the blocklists. But, we can release the domains extracted from Certificate Transparency logs, which are already public. In the paper, we provide the specific collection dates from each source so that interested researchers can download the data or request access from the original data sources.

References

[1] abuse.ch. URLhaus - Sharing malicious URLs that are being used for malware distribution, 2026. https://urlhaus.abuse.ch/api/.

[2] AlienVault. Open Threat Exchange (OTX). https://otx.alienvault.com/browse/global/pulses, 2026.

[3] Mauro Allegretta, Marco Gramaglia, Giuseppe Siracusano, Roberto Gonzalez, and Juan Caballero. Web of Shadows: Investigating Malware Abuse of Internet Services. *Computers & Security*, 149(C), 2025.

[4] Mauro Allegretta, Giuseppe Siracusano, Roberto Gonzalez, and Marco Gramaglia. Are crowd-sourced CTI datasets ready for supporting anti-cybercrime intelligence? *Computer Networks*, 234, 2023.

[5] Eihal Alowaisheq, Peng Wang, Sumayah Alrwais, Xiaojing Liao, XiaoFeng Wang, Tasneem Alowaisheq, Xianghang Mi, Siyuan Tang1, and Baojun Liu. Cracking the Wall of Confinement: Understanding and Analyzing Malicious Domain Take-downs. In *Network and Distributed System Security Symposium (NDSS)*, 2019.

[6] Joshua Anderson. FreeDNS - Free DNS hosting, 2026. https://freedns.afraid.org/.

[7] Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. Building a dynamic reputation system for DNS. In *USENIX Security Symposium (USENIX Security)*, 2010.

[8] Appiculous UG. Short Menu - URL Shortener, Branded Links, Analytics. https://shortmenu.com/, 2026.

[9] APWG. eCrime eXchange (eCX), 2025. https://apwg.org/ecx.

[10] ArchivePH. Webpage archive, 2026. https://archive.ph/.

[11] Paper artifacts. https://zenodo.org/records/20284621, 2026.

[12] Tim Behnke. Blocking the entire domain s3-eu-central-1.ionoscloud.com (81.173.115.248), 2024. https://forums.malwarebytes.com/topic/307585-blocking-the-entire-domain-s3-eu-central-1ionoscloudcom-81173115248/.

[13] Leyla Bilge, Sevil Sen, Davide Balzarotti, Engin Kirda, and Christopher Kruegel. Exposure: A passive DNS analysis service to detect and report malicious domains. *ACM Transactions on Information and System Security*, 16(4), 2014.

[14] Bitly. Bitly - Shorten, Share, and Grow Your Links. https://bitly.com/, 2026.

[15] Stéphane Bortzmeyer and Shumon Huque. RFC 8020: NXDOMAIN: There Really Is Nothing Underneath, 2016.

[16] Xander Bouwman, Harm Griffioen, Jelle Egbers, Christian Doerr, Bram Klievink, and Michel Van Eeten. A Different Cup of TI? The Added Value of Commercial Threat Intelligence. In *USENIX Security Symposium (USENIX Security)*, 2020.

[17] Xander Bouwman, Victor Le Pochat, Pawel Foremski, Tom Van Goethem, Carlos H. Ganan, Giovane C. M. Moura, Samaneh Tajalizadehkhoob, Wouter Joosen, and Michel van Eeten. Helping Hands: Measuring the Impact of a Large Threat Intelligence Sharing Community. In *USENIX Security Symposium (USENIX Security)*, 2022.

[18] Frank Cangialosi, Taejoong Chung, David Choffnes, Dave Levin, Bruce M. Maggs, Alan Mislove, and Christo Wilson. Measurement and Analysis of Private Key Sharing in the HTTPS Ecosystem. In *ACM Conference on Computer and Communications Security (CCS)*, 2016.

[19] Alejandro Celaya. shlinkio/shlink: The definitive self-hosted URL shortener. https://github.com/shlinkio/shlink, 2026.

[20] Euijin Choo, Mohamed Nabeel, Doowon Kim, Ravindu De Silva, Ting Yu, and Issa Khalil. A Large Scale Study and Classification of VirusTotal Reports on Phishing and Malware URLs. *ACM on Measurement and Analysis of Computing Systems*, 7(3), 2023.

[21] Cloudflare Radar. Cloudflare Radar: Top 1 Million Domains. https://developers.cloudflare.com/radar/investigate/domain-ranking-datasets/, 2026.

[22] Common Crawl. Common Crawl - Open Repository of Web Crawl Data. https://commoncrawl.org/, 2026.

[23] DealFuel. Targeter Pro - Link Shortener | Lifetime Access. https://www.dealfuel.com/seller/targeter-pro-lifetime-access/, 2026.

[24] DealFuel. Zat.ink - URL Shortening Tool - Annual Access | DealFuel. https://www.dealfuel.com/seller/zat-ink-url-shortening-tool/, 2026.

[25] Fatih Deniz, Mohamed Nabeel, Ting Yu, and Issa Khalil. MANTIS: Detection of Zero-Day Malicious Domains Leveraging Low Reputed Hosting Infrastructure. In *IEEE Symposium on Security and Privacy (S&P)*, 2025.

[26] Department of Homeland Security. Joint DHS-DOJ "Operation Protect Our Children" Seizes Website Domains Involved in Advertising and Distributing Child Pornography, 2011. https://www.dhs.gov/archive/news/2011/02/15/joint-dhs-doj-operation-protect-our-children-seizes-website-domains-involved.

[27] Ernesto Van dewar Sar. U.S. Government Shuts Down 84,000 Websites, 'By Mistake', 2011. https://torrentfreak.com/u-s-government-shuts-down-84000-websites-by-mistake-110216/.

[28] Dub.co, Inc. Open-source link management infrastructure, 2026. https://github.com/dubinc/dub.

[29] Álvaro Feal, Pelayo Vallina, Julian Gamba, Sergio Pastrana, Antonio Nappa, Oliver Hohlfeld, , Narseo Vallina-Rodriguez, and Juan Tapiador. Blocklist Babel: On the Transparency and Dynamics of Open Source Blocklisting. *IEEE Transactions on Network and Service Management*, 18(2), 2021.

[30] Ian Foster. DNS Coffee. https://dns.coffee/, 2026.

[31] GigaThemes. Adlinkfly - monetized url shortener. https://gigathemes.club/descarga/adlinkfly-monetized-url-shortener/, 2026.

[32] Google. Certificate Transparency, 2026. https://certificate.transparency.dev/.

[33] Google. Chrome User Experience Report (CrUX). https://developer.chrome.com/docs/crux/, 2026.

[34] Google. VirusTotal, 2026. https://virustotal.com.

[35] ICANN Security and Stability Advisory Committee (SSAC). DNS Blocking: Benefits Versus Harms, 2011. https://itp.cdn.icann.org/en/files/security-and-stability-advisory-committee-ssac-reports/sac-050-en.pdf.

[36] ICANN Security and Stability Advisory Committee (SSAC). SAC 127: DNS Blocking Revisited, 2025. https://itp.cdn.icann.org/en/files/security-and-stability-advisory-committee-ssac-reports/sac127-dns-blocking-revisited-16-05-2025-en.pdf.

[37] Short.cm Inc. URL Shortener - Free Link Shortener with Custom Domains | Short.io. https://short.io/, 2026.

[38] Beomjin Jin, Eunsoo Kim, Hyunwoo Lee, Elisa Bertino, Doowon Kim, and Hyoungshick Kim. Sharing Cyber Threat Intelligence: Does it Really Help? In *Network and Distributed System Security Symposium (NDSS)*, 2024.

[39] KOR Labs. Subdomain Providers: Collection of Subdomain Service Providers, 2023. https://github.com/korlabsio/subdomain_providers.

[40] KOR Labs. URL Shortener: Collection of URL Shortening Services, 2023. https://github.com/korlabsio/urlshortener.

[41] David Kravets. Secret Service Seizes JotForm.com, Nuking Millions of Online Forms, 2012. https://www.wired.com/2012/02/jotform-domain-seizure/.

[42] Bryan Krebs. Microsoft Disrupts 'Nitol' Botnet in Piracy Sweep, 2012. https://krebsonsecurity.com/2012/09/microsoft-disrupts-nitol-botnet-in-piracy-sweep/.

[43] Marc Kührer, Christian Rossow, and Thorsten Holz. Paint it Black: Evaluating the Effectiveness of Malware Blacklists. In *International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2014.

[44] Udesh Kumarasinghe, Mohamed Nabeel, and Charitha Elvitigala. Blocklist-Forecast: Proactive Domain Blocklisting by Identifying Malicious Hosting Infrastructure. In *International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2024.

[45] John Kurowski. tldextract. https://pypi.org/project/tldextract/, 2025.

[46] Corben Leo. getallurls (gau). https://github.com/lc/gau, 2026.

[47] Vector Guo Li, Matthew Dunn, Paul Pearce, Damon McCoy, Geoffrey M Voelker, and Stefan Savage. Reading the tea leaves: A comparative analysis of threat intelligence. In *USENIX Security Symposium (USENIX Security)*, 2019.

[48] Linkvertise. Linkvertise | The Search Engine for exclusive Content. https://linkvertise.com/, 2026.

[49] Daiping Liu, Zhou Li, Kun Du, Haining Wang, Baojun Liu, and Haixin Duan. Don't Let One Rotten Apple Spoil the Whole Barrel: Towards Automated Detection of Shadowed Domains. In *ACM Conference on Computer and Communications Security (CCS)*, 2017.

[50] Mingxuan Liu, Yiming Zhang, Xiang Li, Chaoyi Lu, Baojun Liu, Haixin Duan, and Xiaofeng Zheng. Understanding the Implementation and Security Implications of Protective DNS Services. In *Network and Distributed System Security Symposium (NDSS)*, 2024.

[51] Majestic. Majestic Million. https://majestic.com/reports/majestic-million, 2026.

[52] Malicia Lab. iocsearcher. https://github.com/malicialab/iocsearcher, 2026.

[53] Sourena Maroofi, Maciej Korczyński, Cristian Hesselman, Benoit Ampeau, and Andrzej Duda. Comar: Classification of Compromised versus Maliciously Registered Domains. In *European Symposium on Security and Privacy (EuroS&P)*, 2020.

[54] Christian Martorella. theHarvester. https://github.com/laramies/theHarvester, 2026.

[55] Andy Maxwell. Google Drive Blackout in Italy After Another Major Anti-Piracy Blunder, 2024. https://torrentfreak.com/google-drive-blackout-in-italy-after-another-major-anti-piracy-blunder-241020/.

[56] Microsoft. Azure blob storage | microsoft azure, 2025. https://azure.microsoft.com/en-us/products/storage/blobs.

[57] mmnemonic. Argus Managed Defence. https://passivedns.mnemonic.no/, 2026.

[58] Mozilla. Public Suffix List, 2026. https://publicsuffix.org/.

[59] Alexander Tobias Neumann, Johannes Barnickel, and Ulrike Meyer. Security and Privacy Implications of URL Shortening Services. In *Workshop on Web 2.0 Security and Privacy (W2SP)*, 2010.

[60] Palin Ningthoujam. URL Toolbox: 90+ URL Shortening Services, 2008. https://mashable.com/archive/url-shortening-services.

[61] Zul Odgerel, Yevheniya Nosyk, Jan Bayer, Sourena Maroofi, Louis Bedeschi, Andrzej Duda, and Maciej Korczynski. Short Path to Phishing: Identifying Misused URL Shortening Services in the Wild. In *APWG Symposium on Electronic Crime Research (eCrime)*, 2025.

[62] Adam Oest, Yeganeh Safaei, Adam Doupé, Gail-Joon Ahn, Brad Wardman, and Kevin Tyers. PhishFarm: A Scalable Framework for Measuring the Effectiveness of Evasion Techniques against Browser Phishing Blacklists. In *IEEE Symposium on Security and Privacy (S&P)*, 2019.

[63] Adam Oest, Yeganeh Safaei, Penghui Zhang, Brad Wardman, Kevin Tyers, Yan Shoshitaishvili, and Adam Doupé. PhishTime: Continuous Longitudinal Measurement of the Effectiveness of Anti-Phishing Blacklists. In *USENIX Security Symposium (USENIX Security)*, 2020.

[64] OpenPhish. OpenPhish - Phishing Intelligence. https://openphish.com/, 2026.

[65] OWASP. OWASP Amass Project, 2025. https://github.com/owasp-amass/amass/.

[66] David Piscitello. The value of assessing collateral damage before requesting a domain seizure, January 2012. https://www.icann.org/en/system/files/files/seizure-collateral-assess-24jan13-en.pdf.

[67] Andreas Pitsillidis, Chris Kanich, Geoffrey M Voelker, Kirill Levchenko, and Stefan Savage. Taster's Choice: A Comparative Analysis of Spam Feeds. In *Proceedings of the Internet Measurement Conference (IMC)*, 2012.

[68] Carlos Polop. MalwareWorld: A project based on +500 blacklists, 2026. https://www.malwareworld.com/textlists/suspiciousDomains.txt.

[69] CERT Polska. Warning List of Malicious Domains, 2026. https://cert.pl/en/warning-list/.

[70] Spamhaus Project. SpamHaus Intelligence API, 2026. https://www.spamhaus.com/data-access/intelligence-api/.

[71] projectdiscovery team. Subfinder: Fast Passive Subdomain Enumeration Tool, 2026. https://github.com/projectdiscovery/subfinder.

[72] projectdiscovery team. URLFinder: Passive URL Discovery Tool, 2026. https://github.com/projectdiscovery/urlfinder.

[73] Steve Ragan. Microsoft's takedown of No-IP pushes innocents into the crossfire, 2014. https://www.csoonline.com/article/547636/business-continuity-microsofts-takedown-of-no-ip-pushes-innocents-into-the-crossfire.html.

[74] Sivaramakrishnan Ramanathan, Jelena Mirkovic, and Minlan Yu. BLAG: Improving the Accuracy of Blacklists. In *Network and Distributed System Security Symposium (NDSS)*, 2020.

[75] Rebrandly. Free link shortener for branded urls, analytics & qr codes. https://www.rebrandly.com/, 2026.

[76] RevoltSecurities. Subdominator - Unleash the Power of Subdomain Enumeration, 2026. https://github.com/RevoltSecurities/Subdominator.

[77] Doyen Sahoo, Chenghao Liu, and Steven CH Hoi. Malicious URL Detection using Machine Learning: A Survey. *arXiv preprint arXiv:1701.07179*, 2017.

[78] Silvia Sebastián, Raluca-Georgia Diugan, Juan Caballero, Iskander Sanchez-Rola, and Leyla Bilge. Domain and Website Attribution beyond WHOIS. In *Annual Computer Security Applications Conference (ACSAC)*, 2023.

[79] Ravindu De Silva, Mohamed Nabeel, Charith Elvitigala, Issa Khalil, Ting Yu, and Chamath Keppitiyagama. Compromised or Attacker-Owned: A Large Scale Classification and Study of Hosting Domains of Malicious URLs. In *USENIX Security Symposium (USENIX Security)*, 2021.

[80] Sushant Sinha, Michael Bailey, and Farnam Jahanian. Shades of Grey: On the effectiveness of reputation-based "blacklists". In *IEEE International Conference on Malicious and Unwanted Software (MALWARE)*, 2008.

[81] Marco Squarcina, Mauro Tempesta, Lorenzo Veronese, Stefano Calzavara, and Matteo Maffei. Can I Take Your Subdomain? Exploring Same-Site Attacks in the Modern Web. In *USENIX Security Symposium (USENIX Security)*, 2021.

[82] Cisco Talos Intelligence Group (Talos). PhishTank: Join the fight against phishing, 2026. https://phishtank.org/.

[83] thedevs-network. Kutt.to. https://github.com/thedevs-network/kutt, 2026.

[84] Kurt Thomas, Rony Amira, Adi Ben-Yoash, Ori Folger, Amir Hardon, Ari Berger, Elie Bursztein, and Michael Bailey. The Abuse Sharing Economy: Understanding the Limits of Threat Exchanges. In *International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2016.

[85] Ye Tian, Yanqiu Yu, Jianguo Sun, and Yanbin Wang. From past to present: A survey of malicious URL detection techniques, datasets and code repositories. *Computer Science Review*, 58, 2025.

[86] TinyURL LLC. https://tinyurl.com/, 2026.

[87] TomNomNom. waybackurls. https://github.com/tomnomnom/waybackurls, 2022.

[88] Tranco. Tranco: A Research-Oriented Top Sites Ranking, 2026. https://tranco-list.eu/download_daily/9WNY2.

[89] Zisis Tsiatsikas, Georgios Karopoulos, and Georgios Kambourakis. Measuring the Adoption of TLS Encrypted Client Hello Extension and its Forebear in the Wild. In *European Symposium on Research in Computer Security (ESORICS)*, 2022.

[90] Université Toulouse Capitole. URL Phishing Domain List, 2025. https://dsi.ut-capitole.fr/blacklists/download/phishing.tar.gz.

[91] Université Toulouse Capitole. URL Shortener Domain List, 2025. https://dsi.ut-capitole.fr/blacklists/download/shortener.tar.gz.

[92] Vector Guo Li and Matthew Dunn and Paul Pearce and Damon McCoy and Geoffrey M. Voelker and Stefan Savage. Reading the Tea leaves: A Comparative Analysis of Threat Intelligence. In *USENIX Security Symposium (USENIX Security)*, 2019.

[93] Thomas Vissers, Peter Janssen, Wouter Joosen, and Lieven Desmet. Assessing the Effectiveness of Domain Blacklisting Against Malicious DNS Registrations. In *IEEE Security and Privacy Workshops*, 2019.

[94] Internet Archive WayBack Machine, 2026. https://web.archive.org/.

[95] WeaDown. Premium URL Shortener. https://weadown.com/res/premium-url-shortener/, 2026.

[96] Mingxuan Yao, Jonathan Fuller, Ranjita Pai Kasturi, Saumya Agarwal, Amit Kumar Sikder, and Brendan Saltaformaggio. Hiding in Plain Sight: An Empirical Study of Web Application Abuse in Malware. In *USENIX Security Symposium (USENIX Security)*, 2023.

[97] YOURLS Project. Your Own URL Shortener. https://yourls.org/, 2026.

[98] Zesle Software Inc. T2M URL Shortener. https://t2mio.com/, 2025.

Source	Apexes	Excl.	Subdomains
Crt.sh	43,303 (50.6%)	1,158	2,634,283
THC	31,562 (36.9%)	309	5,382,110
Chaos	29,861 (34.9%)	222	79,608,490
VT API	26,546 (31.0%)	2,492	1,491,181
VT2	20,208 (23.6%)	3,857	2,950,511
WaybackArchive	18,238 (21.3%)	376	4,936,478
CTL2	16,861 (19.7%)	849	45,305,671
RapidDNS	16,476 (19.2%)	59	4,170,935
CTL1	11,935 (13.9%)	490	35,216,017
VT1	9,199 (10.7%)	222	5,269,007
HudsonRock	4,921 (5.7%)	10	17,512
Digitorus	4,027 (4.7%)	1	11,868
ThreatCrowd	3,397 (4.0%)	30	118,666
HackerTarget	1,480 (1.7%)	-	88,621
Anubis	723 (0.8%)	-	737,708
CommonCrawl	81 (<0.1%)	-	2,724
SiteDossier	21 (<0.1%)	-	223
CertSpotter	-	-	-
Chinaz	-	-	-
DNSDumpster	-	-	-
DNSRepo	-	-	-
PugRecon	-	-	-
Robtex	-	-	-
RSECloud	-	-	-

Table 13: Coverage of the subdomain enumeration sources used: number of apexes with at least one subdomain, number of exclusive apexes only covered by that source, and total number of subdomains enumerated. Seven sources did not enumerate any subdomains.

[99] Chaoyi Zha. Polr. `https://polrproject.org/`, 2025.

[100] Jing Zhang, Ari Chivukula, Michael Bailey, Manish Karir, and Mingyan Liu. Characterization of Blacklists and Tainted Network Traffic. In *International Conference on Passive and Active Network Measurement (PAM)*, 2013.

[101] Johannes Zirngibl, Steffen Deusch, Patrick Sattler, Juliane Aulbach, Georg Carle, and Mattijs Jonker. Domain Parking: Largely Present, Rarely Considered! In *Traffic Monitoring and Analysis (TMA)*, 2022.

A Enumeration Sources Coverage

Tables 13 and 14 summarize the contribution of the subdomain and URL enumeration sources, respectively. For each source, the tables show the number of apexes for which at least one resource (subdomain or URL) was returned, the number of exclusive apexes for which that source was the only one providing resources, and the total number of resources enumerated. The percentages are computed across all apexes evaluated including the GT and evaluation dataset, totaling 85,571 apexes.

Source	Apexes	Excl.	URLs
WaybackArchive	27,696 (32.4%)	2,919	414,216,453
AlienVault OTX	26,695 (31.2%)	2,923	429,407
URLScan	20,311 (23.7%)	793	792,360
VT API	20,046 (23.4%)	1,532	330,504
VT2	11,933 (13.9%)	78	843,477
CommonCrawl	-	-	-

Table 14: Coverage of the URL enumeration sources used. CommonCrawl did not return any URLs due to a bug in URLFinder.

For subdomain enumeration, the largest coverage is provided by crt.sh with 43,303 (50.6%) apexes, followed by THC with 31,562 (36.9%), Chaos with 29,861 (34.9%), and the VirusTotal (VT) API with 26,546 (31.0%). The VT API and the VT2 dataset are the sources providing the most exclusive coverage. Chaos and CTL2 provide the most subdomains, although from a smaller number of apexes. The fact that the most complete source only covers roughly half of the apexes illustrates the need to combine multiple sources. Still, of the 24 sources, seven did not produce any subdomains (i.e., CertSpotter, Chinaz, DNSDumpster, DNSRepo, PugRecon, Robtex, RSECloud) and another seven (i.e., HudsonRock, Digitorus, ThreatCrowd, HackerTarget, Anubis, CommonCrawl, SiteDossier) provide little information not available from the other sources. Those 14 sources could be dropped to increase the enumeration speed.

For URL enumeration, WayBackArchive provides more than two orders of magnitude more URLs than the other sources thanks to its archival nature. AlienVault OTX contributes the most exclusive apexes, likely due to its focus on malicious domains, while URLScan and the VT API also contribute a significant number of exclusive apexes. CommonCrawl never returned any URLs due to a bug in URLFinder's handling of this source. We have reported the issue to URLFinder. Among the sources with results, the VT2 dataset has the lowest coverage and exclusive apexes, but it is the most efficient as it does not require querying an API. We conclude that CommonCrawl should be removed until the bug is fixed, but the other five URL enumeration sources should be kept.

Overall, no single source provides large coverage by itself. Thus, effective enumeration requires combining multiple sources. But, 14 out of 24 subdomain enumeration sources and one out of six URL enumeration sources could be removed with little impact to speed up BlockMeNot.