Indicator of Benignity: An Industry View of False Positive in Malicious Domain Detection and its Mitigation

Daiping Liu*, Danyu Sun†, Zhenhua Chen*, Shu Wang* and Zhou Li†
*Palo Alto Networks, Inc. †University of California, Irvine
{dpliu, zchen, shuwang}@paloaltonetworks.com, {danyus2, zhou.li}@uci.edu

***Abstract*—Malicious domain detection serves as a critical technique to keep users safe against cyber attacks. Although these systems have demonstrated remarkable detection capabilities, the magnitude of their false positives (FPs) in the real world remains unknown and is often overlooked. To shed light on this essential aspect, we conduct the first measurement study using 6-year FP reports collected from one of the largest global cybersecurity vendors. Our findings reveal that the popularity-based top domain lists that are commonly adopted by current detection systems are insufficient to avoid FPs. In fact, there are still a non-trivial number of FPs in production. We posit that one of the main reasons is that efforts in this area have predominantly focused on detecting malicious indicators, *i.e.,* Indicator of Compromise (IOC), and have made light of the benign ones, *i.e., Indicator of Benignity (IOB)*.**

Invthis paper, we make the first effort focusing on IOB detection. Our work is built upon our key finding that for many FPs in production, their IOBs can be found on the Internet. However, due to the openness of the Internet and unstructured Web content, we face two main challenges to identify these IOBs: understanding what an IOB is and assessing the trustworthiness of an IOB. To address these challenges, we propose a *transitive trust model* for IOB and implement it in a system called `IOBHunter`. `IOBHunter` leverages LLM and chain-of-thought (CoT) which have demonstrated promising capabilities to address several other security threats. Our evaluation using a dataset that contains verified FPs shows that `IOBHunter` can achieve 99.22% precision and 68.6% recall. `IOBHunter` is further evaluated in a two-months real-world deployment, in which `IOBHunter` has identified 4,338 confirmed FPs and 2,051 compromised domains.

I. Introduction

DNS domains have been constantly abused by attackers for illicit activities. For instance, malware encodes data in domain names for command and control (C2), and scammers exploit domains that resemble well-known legitimate ones for phishing attacks. To detect malicious domains, many methods have been proposed in the past [21] [25] [22] [37] [71] [45] [23] [66] [53] [61]. These detection systems have demonstrated promising capabilities and accuracy in detecting malicious domains in their evaluations. Unfortunately, we still lack an in-depth understanding of the magnitude and impact of false positives (FPs) in large-scale longitudinal real-world deployments. Considering that fear of FPs is one of the main concerns for these systems to be adopted in production [63], more effort should be put into understanding the characteristics of their FPs and investigating how to further reduce FPs.

Measurement Study. In this work, we conduct the first large-scale measurement study of FPs of malicious domain detectors. The primary challenge of such a study lies in collecting data and determining ground truth. To address this challenge, our study relies on FPs reported by users of a security vendor ***SV***. ***SV*** has built and deployed tens of malicious domain detectors that detect 1.6M new malicious domains from ∼7B DNS queries on average per day. These malicious domains are used by firewalls of more than 65K organizations around the world. Whenever users notice potential FPs on their firewalls, they can report to ***SV*** whose security researchers manually investigate and decide whether an FP report is correct (*i.e., Accepted FP*) or not (*i.e., Rejected FP*). An additional unique advantage of user-reported FPs is that users often provide a justification of benignity, so that we can gain deeper insight into why FPs are generated. Considering that users of ***SV*** are mostly Security Operations Center analysts of enterprises, their justification along with manual verification by ***SV*** researchers makes these FP reports a dataset with reasonable ground truth.

During 2019∼2024, users have reported 123,491 FPs, with 121,073 for 118,093 unique fully qualified domain names (FQDNs) accepted and 2,418 for 2,022 unique FQDNs rejected. Our analysis of this dataset reveals several interesting findings. First, half FPs are reported within 120 days after domains are detected as malicious. Thus, to evaluate the FP rate, it is more reasonable to deploy detectors in production for more than 4 months. Second, FPs in production have a long tail distribution, with 97.7% FQDNs in FPs being reported only once by one user. Most of these FQDNs are under unique root domains. As a result, FP mitigation approaches need to be generic to cover a diverse set of benign FQDNs that could become FPs. We further find that current popularity-based top lists that are commonly adopted by detectors, such as Tranco [62], cannot effectively mitigate FPs. In particular, they can only cover at most ∼38% FPs. Therefore, more effort is required to identify better generic ways to mitigate FPs in production. Finally, for ∼55% FPs, benign indicators of the

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.241869
www.ndss-symposium.org

detected domains can be found on the Internet. This may be because such FPs are more likely to be noticed and reported by users. Hence, a promising direction to reduce FPs is to proactively identify IOBs from the Internet.

`IOBHunter`. Motivated by the findings in our measurement study, we make the first effort to automatically collect IOBs for domains on the Internet. To achieve this, we face two main challenges. First, Web content is mostly unstructured and contains diverse information about an FQDN. We need to separate the wheat from the chaff and identify valid IOBs. Second, the Internet is an open space and any user, including malicious actors, can publish arbitrary content. It is critical to assess the trustworthiness of collected IOBs.

To address these challenges, we propose a *transitive trust model* for IOB, which is built on the principle that a domain is benign if it is owned by a trusted owner or certified by a trusted source. Specifically, the transitive trust model addresses the two challenges with a clear definition of IOB and an iterative search-and-check schema for the assessment of trustworthiness, respectively. Given a target domain, we first leverage search engines to obtain websites that provide valid IOBs (*i.e.,* matching our IOB definition). These websites are referred to as IOB sources. Then, we further check if these IOB sources are root of trust or certified by other trusted sources based on a set of trust transit policies. This search-and-check process is performed iteratively until the root of trust is reached or no new result is returned by search engines. Our transitive trust model currently considers *government organizations* to be the only root of trust that has served as the root of trust in many other scenarios in our daily life.

We implement the transitive trust model in a system called `IOBHunter`. `IOBHunter` leverages large language models (LLM) and chain-of-thought (CoT) which have shown promising capabilities in other security problems, such as detection of harmful online content [38] [74] [75] [83], phishing websites [55] [55], and cyber threat intelligence (CTI) [59] [69] [40] [28]. To our knowledge, `IOBHunter` is the first system that applies LLMs and CoT to automated IOB collection.

To demonstrate the efficacy of `IOBHunter`, we perform evaluations using the FP CR dataset as ground truth. Our evaluation results show that `IOBHunter` achieves 99.22% precision and 68.6% recall when GPT-4o is used. In addition, evaluations using two other popular LLM models, Gemini2.5-flash and Qwen3-32B, show that `IOBHunter` using different models can achieve comparable accuracy. The relatively low recall is mainly because many IOBs come from untrusted sources such as social media platforms. While these IOBs are mostly valid, they can be easily manipulated, especially when attackers are aware of `IOBHunter`.

`IOBHunter` is further evaluated on a dataset containing all malicious domains detected by ***SV*** from 2025-03-01 to 2025-05-01. Among the 15M detected FQDNs, `IOBHunter` finds trusted IOBs for 6,571. After manual investigation and discussion with ***SV*** researchers, 4,338 are confirmed to be FPs and ***SV*** *researchers have flipped the verdicts of these FQDNs from malicious to benign*. In addition, for 2,051 cases, the IOBs identified by `IOBHunter` are true. However, there are also confirmed malicious activities associated with these FQDNs. Thus, these FQDNs are classified as compromised. Both types can be considered correct detections by `IOBHunter`. Finally, the IOBs for 182 cases are incorrect and are mainly caused by LLM hallucinations [81].

Data Availability. To motivate and support future research on IOB, we make a partial of our FP dataset available. To comply with our company's policies and legal requirements, we can share FP cases that were only detected by third-party feeds and have been flipped to benign by third-parties (*i.e.,* considered as FPs by third-parties, too). Also, since users' comments in FP reports can contain sensitive and personally identifiable information, they cannot be shared. The dataset is available at https://github.com/dpliu/iobhunter-dataset.

II. Background

This section first provides an overview of the domain system. Then, we demonstrate the scale of FPs generated by malicious domain detectors deployed by a security vendor, which motivates this work. Next, we elaborate the types of malicious domain detectors deployed by the vendor and the allowlists used by the vendor for FP mitigation.

This work focuses on *domains*, as malicious domain detection has been a hot topic in recent decades for both academia [63] [21] [25] [22] and industry [1] [2]. Other types of entities, such as URLs and IPs, are out of scope.

A. Basics of Domain Names

A **domain name** (or **domain**) is a human-friendly representation of a resource accessible through the Internet. When a domain ends with a dot "." (*e.g.,* `www.foo.com.`), it is referred to as a Fully Qualified Domain Name (**FQDN**). The rightmost label in an FQDN (*e.g.,* `com` in `www.foo.com.`) is called a top-level domain (**TLD**). Next to TLD is the second-level domain (**2LD**). Some 2LDs (*e.g.,* `foo.com.`) can be bought from registrars (like GoDaddy) by the public, while others (*e.g.,* `co.uk.`) are operated by registrars and public users can only purchase 3LDs under these 2LDs from registrars. In this work, we use effective TLDs (**eTLDs**) to collectively refer to TLDs, 2LDs, 3LDs, *etc.* that are operated directly by registrars. Domains with one more label than eTLDs (*e.g.,* `foo.co.uk`) are called **root domains**. In addition, domains with one or more labels than a root domain are called **subdomains** of the root domain. Users can host Web content and services on a URL under a domain, *e.g.,* `www.foo.com/account`. By default, subdomains are managed by the same owner as the root domain. However, root domain owners can also delegate subdomains to others (*e.g.,* a GitHub user can create a website under a subdomain of `github.io.`).

B. Motivation and Problem Statement

This study was motivated by our observations in a production environment at a major security vendor (denoted as ***SV***), which provides cybersecurity solutions globally. ***SV*** has deployed tens of malicious domain detectors in production,

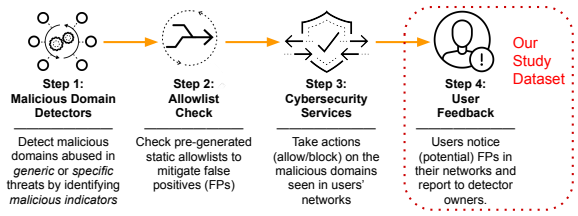

Fig. 1: Typical deployment of malicious domain detectors.

following the common workflow illustrated in Figure 1. These detectors detect *1.6M new malicious domains* from ∼7B DNS queries on average per day, and the detected domains are integrated into the firewall policies by more than 65K organizations around the world. False positive (FP) is a main concern of ***SV***'s users, and ***SV*** handles FPs reported by the users (Step 4 in Figure 1) with extensive manual analysis (elaborated in §III-A). Each FP will be classified into **Accepted FP** (*i.e.,* the domain should *not* be alerted) and **Rejected FP** (*i.e.,* the domain is indeed malicious) after investigation.

Although ***SV***'s detectors have been tuned towards a lower FP rate and a few allowlists[1] have been used (Step 2 in Figure 1), there are still *∼20K FPs* reported per year on average during 2019∼2024, *98%* of which were determined to be accepted FPs. These accepted FPs occasionally resulted in financial losses on ***SV***'s customers.

Problem statement. Understanding the nature of user-reported FPs is essential to guide continuous improvement of detectors and mitigate customer losses. However, to our knowledge, no prior measurement study has been done for this important problem. This motivates us to conduct the *first* large-scale study using the six-year FP data from ***SV***.

Note that this work does not pay special attention to one detector. For instance, we do not present the detection accuracy of an individual detector or compare the performance across detectors. Instead, we intend to study the *generic* characteristics of FPs triggered by various production-grade detectors and explore approaches for effective FP mitigation *universally*. Our objective is to answer the following research questions:

- **RQ1.** What are the statistics of FPs reported by users? (§III-B1)
- **RQ2.** What are the characteristics of the user-reported FPs? (§III-B3§III-B4§III-B5)
- **RQ3.** Can popularity-based allowlists be tuned to further reduce FPs in production? (§III-B2)
- **RQ4.** How do researchers investigate FP reports (§III-B6)? Can we automate the investigation (§IV)?

C. Malicious Domain Detection

The concrete answers to the above questions highly depend on how malicious domains are detected. Therefore, in this section, we give an overview of the detectors deployed by ***SV***, which is Step 1 in Figure 1. We leave some technical details to Appendix A.

[1]Allowlist is also referred to as whitelist or trust list.

In general, a malicious domain detector can be categorized into three aspects: detection scope, types of malicious indicators, and classification techniques, as summarized in Table I. For detection scope, a detector is either designed based on generic characteristics of malicious domains or specific domain-based attack (*e.g.,* domain squatting). For types of malicious indicators, a detector uses features from domain names themselves or features related to their usage (*e.g.,* domain resolution patterns). For classification techniques, both signature-based and learning-based techniques have been extensively used. In addition to the aforementioned in-house detectors, ***SV*** augments the detection coverage with four widely used third-party feeds, including VirusTotal, Spamhaus, URLhaus, and Abuse.ch.

D. Mitigating FPs with Allowlists

In this section, we survey the common practices for filtering out domain FPs and the solutions deployed by ***SV*** *before* an alert is generated (Step 2 in Figure 1).

Similarly to other security systems such as spam prevention [33], IP blocking [3], and application control [73], malicious domain detectors could leverage pre-generated allowlists to filter out FPs at low cost. The structure and usage of DNS-based allowlists have been documented in RFC 5782 ("DNS Blacklists and Whitelists") [4], although how the list is filled is left to the vendors and developers.

The usual approach is to include the *most popular* domains in allowlists, as they are often registered and used by reputable parties. Before its discontinuation in May 2022, Alexa top list [5] was the de facto standard source of popular domains, which has been used in most research papers [62] and production systems such as Quad9 [6] and Netresec [7]. Yet, Alexa list was vulnerable to rank manipulation [62]. To address this issue, researchers proposed methods to build more reliable lists such as Tranco [62], Secrank [78], whitelst [24], and dynamic allowlists [26]. In fact, after the discontinuation of the Alexa list, Tranco has become the new de facto allowlist, as shown on their website [8]. In addition to public lists, some detectors also build custom allowlists from their own network traffic [66] and include well-known Internet services in allowlists [21] [6].

Allowlists deployed by *SV*. Table II presents how ***SV*** constructs the allowlist. First, domains that consistently appear on the Alexa/Tranco top 1M list in the past 30 days are included in the allowlist. The second source is the top 1M domains ranked from the DNS traffic of ***SV***'s customers. Moreover, domains that provide legitimate services such as content delivery networks (CDN) and web hosting are also included. Finally, ***SV*** uses an allowlist that includes a small set of manually-curated trusted domains, such as those owned by ***SV***'s customers. Note that for domains in the allowlists, some detectors in ***SV*** can still issue alerts on them, *e.g.,* when the website under a domain is compromised.

III. Large-scale Measurement of FPs

The main challenge to studying FPs in production lies in collecting and determining the ground truth. Currently, there

		Example Detectors	Covered by *SV*?
Detection Scope	Generic malicious domains	[21] [25] [22] [57] [49] [37] [43]	✔
	Specific threats		
	Domain squatting	[71] [45] [32]	✔
	Domain generation algorithms	[23] [66]	✔
	DNS rebinding	[42]	✔
	Fast flux	[39]	✔
	Domain shadowing	[53]	✔
	DNS tunneling	[61] [60]	✔
	Malicious websites	[77] [51] [55] [46] [44] [70] [48]	✔
Type of Indicators	Lexical analysis of domain names	[21] [25] [37] [71] [45] [32] [23] [66] [53] [61] [60]	✔
	Domain registration and Whois	[37]	✔
	Domain resolution patterns	[57] [43] [42] [39] [53] [21] [25] [22] [60]	✔
	Malicious web content	[77] [55] [46] [44] [48] [70] [53] [27] [72]	✔
Classification Technique	Supervised machine learning	[21] [25] [22] [37] [53] [66] [53] [77] [51] [72]	✔
	Semi-supervised machine learning	[23] [27]	✔
	Anomaly detection	[60]	✔
	Graph-based analysis	[57] [43] [49]	✔
	Signatures and rules	[46] [44] [70] [48]	✔

TABLE I: Categorization of representative malicious domain detectors.

	Examples	Used by *SV*?
Alexa top domains [5]	[45] [66] [6] *etc.*	✔†
Tranco top domains [62]	[8]	✔
Customer top domains	[66]	✔
Known Internet services	[21] [6]	✔
Other manually trusted domains	N/A	✔

†After the discontinuation of Alexa, ***SV*** switched to Tranco.

TABLE II: Allowlists commonly used by malicious domain detectors to filter out FPs.

is no definitive way to decide if an alerted domain is FP. Otherwise, such an approach would have been used to prevent FPs in advance. As such, our study relies on a dataset of FPs *reported by users* (Step 4 Figure 1). A unique advantage of our dateset is that users often provide a justification of benignity, so we are able to gain deeper insight into why FPs are generated. In addition, from a practical point of view of security services, understanding and mitigating user-reported FPs could result in better security products.

Next, we first describe how user-reported FPs from ***SV*** are collected and processed. Then, we conduct a systematic measurement to understand the characteristics of these FPs. Finally, we discuss the threats to validity of our study.

A. FP Collection

Here, we provide an overview of how ***SV*** handles user-reported FPs. Users can submit **change requests (CR)** for any domain on a Web portal of ***SV***. Each CR includes three required fields: *the domain in question*, the domain's *original category*, and *a new category suggested* by the user for the domain. There is also an optional field, *user comment* where users can provide a description and evidence on why they consider the original category is wrong. Note that the Web portal provides one-way communication, without follow-ups from security researchers to users.

FP CR. Since this work focuses on FPs in production, FP CRs are used in our study. An FP CR is a CR in which the original domain category is malicious and the suggested new verdict is benign. FP CRs are handled by security researchers in ***SV*** who designed and implemented the detectors described in §II-C. An FP CR can be accepted or rejected after manual investigation. For an **accepted FP CR**, the domain category is changed based on users' suggestions and researchers' investigation, so that the domain will no longer be blocked. For a **rejected FP CR**, no action is taken and the domain will remain blocked. Notably, there is a delay between the alert and the CR of a domain. Therefore, the domain under a CR can be truly malicious at the time of detection and became clean at the time of CR submission, which turns the domain into an accepted FP. One such example is a compromised website that gets malicious code removed later.

To review FP CRs, the general guidelines for researchers are that *i)* if there is reliable malicious evidence and the evidence is still valid, reject the FP CR; *ii)* if there is no reliable malicious evidence and benign evidence is provided by users or found on the Internet, accept the FP CR; *iii)* otherwise, decisions are made at researchers' discretion. Basically, malicious evidence includes hosting malicious Web content at researchers' discretion (*e.g.,*, phishing content), connections only from malware (based on VirusTotal and ***SV*** 's in-house detection engine), attribution to attacks in articles by reputable security vendors, and domains resolved to bullet-proof IPs. Benign evidence is mainly a description about a domain's usage for legitimate purposes, as judged by security researchers.

Attributing FP CRs to sources. As described in §II-C, ***SV*** detects malicious domains from two sources, in-house detectors ($\mathcal{D}$) and third-party threat feeds ($\mathcal{F}$). A malicious domain could be detected by both $\mathcal{D}$ and $\mathcal{F}$. To avoid duplicates and simplify the presentation of the results, we attribute each FP CR to one single source. Since we can only decide the detection reasons for the domains detected by $\mathcal{D}$, we choose to attribute a FP CR to $\mathcal{D}$, when it is included by both $\mathcal{D}$ and $\mathcal{F}$. The rule for source attribution can be written more formally as below. 1) An FP CR is attributed to $\mathcal{D}$ as long as the domain

	Accepted FQDN	Accepted Root	Rejected FQDN	Rejected Root
$\mathbb{P}_{\mathcal{D}}$	94,144	89,121	1,329	1,297
$\mathbb{P}_{\mathcal{F}}$	23,949	23,032	693	681
Total Unique	118,093	111,932†	2,022	1,972†

†The total number of unique root domains is less than the sum of $\mathbb{P}_{\mathcal{D}}$ and $\mathbb{P}_{\mathcal{F}}$ because there could be multiple FP CRs for different subdomains under a root domain.

TABLE III: Number of unique FQDNs and root domains in FP CRs.

Root Domain	# of FP CRs	Domain Category
weebly.com	2,125	Web hosting
blogspot.com	146	Web hosting
oastify.com	109	Pentesting tool
smapply.io	61	SaaS service
tripod.com	53	Web hosting
taxesejour.fr	50	Financial services
campuswell.com	45	Education, Health
azurewebsites.net	42	Web hosting
smapply.org	34	SaaS service
readsh101.com	33	Education, Health

TABLE IV: Top 10 root domains in terms of the number of FP CRs received for their subdomains.

is detected by $\mathcal{D}$, and we denote these FP CRs as $\mathbb{P}_{\mathcal{D}}$. 2) An FP CR is attributed to $\mathcal{F}$ if the domain is *only* contained by $\mathcal{F}$, and this set of FP CRs is denoted as $\mathbb{P}_{\mathcal{F}}$.

B. Characteristics of User Reported FPs

1) How many FP CRs in production? (RQ1): In total, 123,491 FP CRs are reported, with 121,073 for 118,093 unique FQDNs being accepted and 2,418 for 2,022 unique FQDNs rejected. The overall statistics suggest that most of the FPs reported by the users are accepted (>98%), underscoring the importance of FP mitigation. Under the attribution scheme in §III-A, Table III presents the breakdowns of the unique FQDNs and root domains in the FP CRs, by sources. We found that most FQDNs (97.7%) receive only one FP CR, revealing a long-tailed distribution of FP FQDNs. Meanwhile, 2,772 FQDNs receive more than one FP CR, with the highest number of FP CRs for a single FQDN being 54. Most of these FQDNs belong to security tools such as pentesting. These tools are used for both legitimate and malicious purposes, and thus their domains are usually kept as malicious. Users keep complaining that these domains are FPs.

To reduce the workload of reviewing FP CRs, one approach can be grouping FP CRs by root domains and processing them by groups. However, we found that most FQDNs are hosted under unique root domains (113,738 unique root domains for 123,491 FP CRs), and Table IV lists the top 10 root domains after grouping. The distribution is also long-tail (except `weebly.com`, all the other root domains have less than 150 FP CRs). Besides, 6 out of the top 10 root domains belong to Web hosting and SaaS service categories, which allow subdomains to be registered by a third party. Therefore, reviewing after domain grouping is unlikely an effective approach.

Interestingly, a prominent portion of the accepted FQDNs from the threat feed $\mathbb{P}_{\mathcal{F}}$ is also observed in $\mathbb{P}_{\mathcal{D}}$. We speculate on two possible reasons. First, these malicious domains are probably independently detected by both $\mathcal{F}$ and $\mathcal{D}$ which use similar detection approaches and thus suffer similar FPs. Second, as one of the largest security vendors, domains detected by in-house detectors of ***SV*** can propagate to and affect third-party feeds such as VirusTotal [82].

Finally, while the rejection rate for FQDNs in $\mathbb{P}_{\mathcal{F}}$ (2.89%) is significantly higher than 1.41% in $\mathbb{P}_{\mathcal{D}}$, this does not mean that the in-house detectors perform worse than the third-party feeds. The notable difference is due to the fact that not all malicious domains from third-party feeds are ingested by ***SV***, and only those more likely to be truly malicious are ingested, as described in Appendix §A.

2) Can popularity-based allowlists be tuned to further reduce FPs? (RQ3): As illustrated in Figure 1, popularity-based allowlists are used to mitigate FPs generated by detectors before alerts are sent to users. Hence, a natural follow-up question is whether we can tune the allowlists to further reduce FPs that will be reported by users. To answer this question, we investigate how many domains in FP CRs appear in public top lists. We use five publicly accessible raw top lists, including Alexa top-1m [5], Majestic Million [9], Cisco Umbrella 1 million [10], Quantcast top 490K websites [11], and Chrome User Experience Report (CrUX) [12]. For each list, we extract the root domains from the FQDNs and match them to the FPs. We did not include Tranco here, as it has merged the other lists like Alexa top-1m.

The first four lists are provided from public websites, and we collect their daily snapshots from 2019-01-01 to 2024-04-01. For Alexa and Quantcast which have been discontinued, the daily lists are collected until their discontinuation dates. Figure 2 shows the percentage of root domains in FP CRs that appear at least once in a public top list. We can see that even if we simply merge all the top lists, only ∼38% root domains could be covered. Even worse, a benign and popular root domain does not necessarily mean that its subdomains are also benign. For example, `weebly.com` is a popular domain. However, it is also abused by attackers to host malicious content on its subdomains. In fact, about 49.2% of the FP CRs are for subdomains whose root domains appear at least once in a public top list.

To further evaluate how likely a larger top list can help, we check the ranking of root domains in FP CRs using CrUX. CrUX provides website ranking based on browser traffic contributed by opt-in Chrome users, and the published ranking is updated monthly. The rank is bucketed on a $log10$ scale with half steps, *e.g.,* top 500, top 1000, *etc.* For each root domain in FP CRs, we get its highest rank bucket from 2019-01 to 2024-12. Figure 3 shows the results. As we can see, about *50%* of the root domains are ranked below 5 million. Considering that the existing 1-million lists have already included many malicious domains [78] [62], if we extend the lists to 5 or even 10 million, it could lead to an unacceptable number of false negatives, and thus dramatically undermine the protection

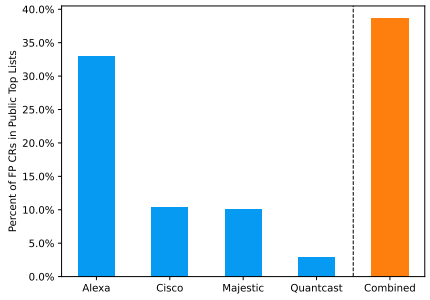

Fig. 2: Percentage of root domains from FP CRs that appear at least once in a public top list.

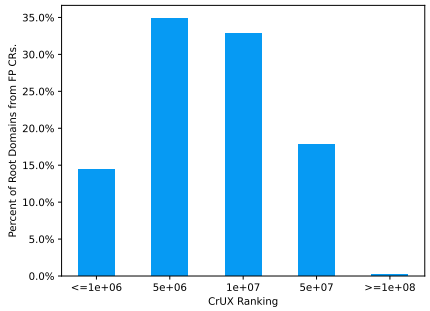

Fig. 3: CrUX ranking of root domains from FP CRs.

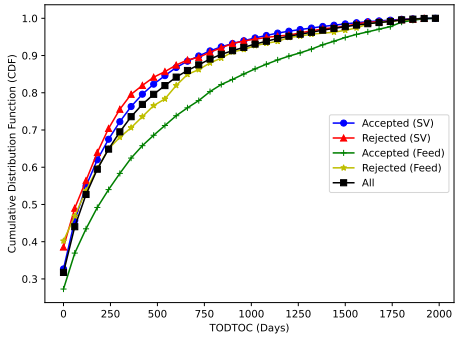

Fig. 4: Distribution of time of detection to time of complaint (TODTOC).

Detection Scope	# of FP CR†	# of Accepted	# of Detections Per Day
Generic detectors	8,518	8,154	∼5K
Domain squatting	830	810	∼8K
DGA	1,130	1,114	∼11M
DNS rebinding	102	64	20
Fast flux	40	39	43
Domain shadowing	0	0	203
DNS tunneling	446	394	13
Malicious websites	87,596	86,401	∼260K

†A domain could be detected by multiple detectors and thus a CR can be counted multiple times under different scopes.

TABLE V: Count of total FP CRs by detection scopes in §II-C. For comparison, the last column lists the number of new detections per day on average by detection scopes.

efficacy in practice.

In summary, it is impractical to mitigate FPs by simply enlarging the allowlists, and complementary approaches should be developed to mitigate FPs in production.

3) Time of detection to time of complaint (RQ2): Next, we count the number of days between when a domain is detected as malicious and when an FP CR is reported for the domain, *i.e., time of detection to time of complaint (TODTOC)*. Figure 4 shows the cumulative distribution of TODTOC. Overall, ∼10% FPs are reported within one week, ∼23% FPs are reported within 30 days and half of FPs are reported within 120 days. Given the long delay of FP reporting, we recommend evaluating a malicious domain detector after more than 4 months deployment, so more than half of FP CRs are likely to be collected to tune the detector.

Then, we compare the distribution of TODTOC by the decision of FP CRs (accepted and rejected) and sources (in-house detectors and third-party feeds). First, we found that FP CRs with smaller TODTOC are more likely to be accepted. This is consistent with the intuition that the earlier an FP is reported, the more likely the FP would cause perceivable negative impacts on users, leading to the correction. Second, there is a prominent difference between the rejected FP from $\mathcal{D}$ and $\mathcal{F}$. This is probably because for domains detected by in-house detectors, researchers often have more evidence on the true-positive domains, and therefore can reject them promptly. However, for domains from public feeds, there is usually no additional context, and thus researchers tend to take longer time for investigating or accepting FP CRs.

4) FP by detection scope (RQ2): Table V presents the count of FP CRs by the detection scopes described in §II-C. We only consider FP CRs by ***SV*** in-house detectors here because we do not know how malicious domains from public feeds are detected. As shown in the table, the detectors of malicious websites dominate the FP CRs. Further investigation reveals that most of these reported domains are owned by small businesses that usually do not have strong security measures to protect their websites. Many of these websites turn out to be compromised at the time of detection and cleaned up at the time of CR submission. As such, users are more likely to report these malicious domains as FPs and the relevant FP CRs are accepted after clean-up. For the other detectors, we found the generic detectors have a much higher FP ratio (over 8K FPs overall with ∼5K daily detection), while the threat-focus detectors like DGA detector (only 1.1K FPs overall with ∼11M daily detection) and domain shadowing detector (0 FPs) are fairly accurate, suggesting it is a better practice to deploy an ensemble of detectors to reduce FPs.

5) FP reasons reported by users (RQ2): We now investigate why a malicious domain is reported as an FP, by inspecting the comments submitted by users in CRs. We use a comment analysis tool developed by ***SV*** researchers, which parses user comments into nine reasons with keyword matching and LLM-

based text analysis. Table VI presents the nine types of comments along with the statistics for each type. Due to the space limit, we present more details about each type along with examples of user comments in Appendix §B.

Among the nine types, three do not provide meaningful information. About 38.4% of the CRs do not have comments. Besides, users simply ask ***SV*** to provide Indicators of Compromise (IOCs) or claim that the domains are clean without providing any benign evidence in 7.2% and 6.7% CR comments, respectively.

The remaining 58,799 CRs contain meaningful information in the comments, which can be categorized into six reason types. The most common reason given by users is that the web content hosted on reported domains is legitimate (`Domain Content`). Unfortunately, using Web content to judge the benignity of a domain is not a safe practice, and we describe two representative cases summarized from comments. First, some users argue a domain is benign as there is no content hosted on the domain (*e.g., "404 error when attempting to reach the site"* and *"the website has absolutely no content"*). Second, some users point out the domain redirects to `google.com` (*e.g., "site redirects to google"* and *"as per our checks url is landing in google"*). Showing no meaningful content or redirection are actually actually common cloaking techniques employed by attackers [80]. As such, we believe to avoid users' confusion, the detector can provide more details about why a domain is detected as malicious, which is especially important when a domain is abused or a website is compromised.

The next common reason is that users need to *use* the domains, *e.g.,* for business purposes (`Valid Use`). In particular, many users mention that they cannot access the services hosted on the reported domains. For example, they cannot send emails or communicate using VoIP systems. Similarly to `Domain Content`, most domains of this reason are detected as malicious because of compromised websites. Therefore, to mitigate collateral damage, it is important for security services to block only malicious traffic to/from the domains without affecting other legitimate services hosted on the same domains, which requires finer-grained detection/blocking capabilities.

Furthermore, for a prominent amount of CRs (10,266), users have checked open-source intelligence (OSINT), such as VirusTotal, AlienVault [13], *etc.*, and claimed that the reported domains are classified as *benign* by them. This reason is particularly interesting for $\mathbb{P}_{\mathcal{F}}$, as the feeds from which these malicious domains are ingested partially overlap with the OSINT mentioned by users. We found that users choose to include OSINT for two main causes. First, some users consider zero detection by an OSINT as benign evidence, while VirusTotal is the only OSINT mentioned by users. However, we found that VirusTotal has a notable number of false negatives compared to the *25* OSINT feeds used by ***SV*** (including VirusTotal). Second, some users consider it benign evidence when $1 \sim 3$ engines on VirusTotal classify a domain as malicious (*e.g., "Only one hit on VT for phishing. Not seeing anything in any run."* and *"my virustotal check shows up 3 hits but this could be false positive"*) However, there is not consensus regarding the threshold of VirusTotal hits [82].

In 8,034 CRs, users claim to own the domains (`User Own`). Most of these FPs are produced by the domain squatting detector, as the FP domain names look similar to the legitimate ones owned by ***SV***'s users but the webpages on the domains are classified as phishing. Though using the registrant information in Whois data to verify domain ownership is likely to mitigate such FPs, unfortunately, due to data privacy policies such as GDPR, more and more domains have anonymized Whois [56]. Deploying domain attribution techniques that do not depend on Whois, *e.g.,* through [67], might be necessary to address this problem.

Although the types of reason in the user comments do not strongly correlate with whether FP CRs are accepted overall, we can see that CRs with the reason `Stale Detection` are all accepted. This result could be relevant to domain registration patterns. In fact, malicious domains are commonly registered for a short period of time, usually one year [37]. Therefore, if a malicious domain was detected years ago, it has probably expired and is no longer used for malicious activities.

*6) Malicious/benign evidence by **SV** researchers (RQ4):* When a security researcher processes a FP CR, evidence needs to be provided to make the final decision, and we study the relation between FP CRs with evidence here. Since ***SV*** does not keep the researchers' justification, including evidence, for accepted and rejected CRs in persistent storage, we instead retrieve the malicious and benign evidence again, following the investigation guidelines of ***SV***.

Malicious evidence for rejected FP CRs. For 2,238 out of 2,418 rejected CRs, we are able to collect two types of malicious evidence. First, the domains in 1,281 CRs were still actively involved in malicious activities at the time of CR submissions, which were confirmed by contacting the detector maintainers. For 1,675 CRs, the domains were mentioned in the intelligence sources maintained by ***SV***, which collected thousands of security blogs and technical articles over years. For 718 CRs, both types of malicious evidence could be found.

Benign evidence for accepted FP CRs. For accepted CRs, we collect two types of benign evidence, including benign indicators on the Internet and convincing benign evidence in user comments. For the first type of evidence, the CR handling system in ***SV*** automatically obtains the websites on the Internet that mention the FQDNs and/or their root domains in the CRs, to aid researchers in CR triaging. Based on our discussion with the researchers, if the website that describes the queried domain is trustworthy and vouches the legitimacy of the domain, the CR are likely to be accepted. Hence, we search the FQDNs associated with the CRs to find the vouching websites as evidence [2]. The second type of benign evidence should either prove user's ownership on the domain (like `User Own` in §III-B5) or domain's valid usage (like `Valid Use` in §III-B5), while both claims can

[2]***SV*** first used the Bing Search API [14] to obtain search results, which was retired on 2025-03-06 [15]. ***SV*** has transitioned to `Gemini Grounding with Google Search` [16]. ***SV*** is also evaluating Google Programmable Search Engine [17].

	Total CRs	No Meaningful Information†			Meaningful Information*					
		No Comment	Ask for IOC w/o Evidence	Claim Clean w/o Evidence	User Own	Valid Use	Clean OSINT	Domain Content	Malware Cleaned	Stale Detection
Accepted $\mathbb{P}_{\mathcal{D}}$	95,024	36,317	6,842	6,487	5,707	14,590	8,621	33,591	1,607	29
Accepted $\mathbb{P}_{\mathcal{F}}$	23,759	8,943	1,826	1,544	2,133	4,546	1,507	8,673	783	20
Rejected $\mathbb{P}_{\mathcal{D}}$	2,818	1,290	188	168	142	246	107	350	10	0
Rejected $\mathbb{P}_{\mathcal{F}}$	1,890	863	67	157	52	104	31	179	6	0

†A comment without meaningful information is classified exclusively into one of the three types.
*A comment with meaningful information can contain multiple types of reasons.

TABLE VI: Statistics of the FP reasons in user comments.

be supported with evidence that can be obtained through Internet search. Based on the above schema for gathering benign evidence, potential benign evidence could be found on the Internet for many accepted CRs (*66,639 out of 121,073*). Meanwhile, 19,375 CRs have convincing benign evidence in user comments, of which 14,749 also have benign evidence on the Internet.

7) Summary of findings: Here we present the prominent lessons we learned from our measurement study.

- **Lesson 1**. User-reported FPs have a long-tail distribution on FQDNs and root domains (§III-B1). Hence, triaging FPs by groups of FQDNs and root domains does not save much manual effort.
- **Lesson 2**. User-reported FPs cannot be effectively mitigated by adding more domains from the popularity-based allowlists like Alexa top-1m. The upper bound of FP mitigation is only ∼38% by allowing all domains from all lists. (§III-B2).
- **Lesson 3**. For ∼55% FPs, benign indicators for the detected domains can be found on the Internet (§III-B4, §III-B5, §III-B6).
- **Lesson 4**. Many FP CRs are related to compromised websites (§III-B5). To better address user complaints, the detectors could explicitly tell users that domains are compromised in these cases.
- **Lesson 5**. Given the trend of anonymizing domain Whois information, using domain ownership as an indicator to mitigate FPs becomes more challenging. A new approach is needed to robustly and effectively attribute domains to owners. (§III-B5)

C. Threats to Validity

As with all measurement studies, there is a risk that our findings may be biased and not representative. Below we discuss the potential issues.

Reliability of FP CR dataset. First, it is likely that not all FPs are reported by users. For example, it is possible that users do not notice an FP. Even if they noticed, they might just add the domains to their local allowlists and do not report to ***SV***. However, we posit that most of these unreported FPs have relatively low impact on users and do not undermine the value of our study. Second, users could submit adversarial FP CRs for truly malicious domains along with misleading comments. Considering that FP CRs are submitted primarily by IT and InfoSec teams of enterprises and organizations that are users of ***SV*** products, the majority of CRs should be trustworthy and adversarial FP CR submissions are rare. Finally, ***SV*** researchers might falsely accept or reject an FP CR. For falsely rejected FPs, users often reach out to support engineers of ***SV*** and open tickets, and we only saw several such cases in our dataset. For falsely accepted FPs, users would need to provide very convincing evidence that a malicious domain is benign (*e.g.,* the domain is registered by a reputable party). We expect such cases to be rare.

Potential bias due to the allowlists used by *SV*. ***SV*** used public and private allowlists shown in Table II, and both might result in detection bias (*e.g.,* other security companies might deploy different allowlists). We argue that the detection variation caused by public lists should be small, as top lists such as Tranco and Alexa are used by nearly every security company based on our discussions with industry partners. The private lists such as the manually-compiled trusted domain list could be different from the ones used by other security companies. However, we observe that private lists mitigate only a small portion of FPs in production. Hence, their impact on the result validity is limited.

IV. Design and Implementation of IOBHunter

A. Motivation & Overview

We are motivated to develop an automated tool to replicate the investigation procedure of security researchers, such that a large ratio of FPs can be mitigated before they bring negative impact on users. According to **Lesson 3** in §III, indicators can be found on the Internet for large amount of accepted FPs, which we term as ***Indicator of Benignity (IOB)***. Therefore, we can proactively collect IOBs for the detected FQDNs and correct/augment the detection results accordingly. IOBs could also reflect the rightful domain owners to address the issue of anonymized Whois (**Lesson 5**). When a domain is alerted because it hosts a compromised website, IOBs could provide necessary context to users (**Lesson 4**). Yet, attackers can forge IOBs, *e.g.,* by creating a website which claims a malicious domain as benign, and we have to select IOBs carefully.

We design IOBHunter to find the *trusted IOBs* on the Internet for the detected domains and aid the alert triaging. Our key innovation is a *transitive trust model* that we develop for finding and asserting the trustworthiness of IOBs. We are inspired by the chain-of-trust verification protocol from Certificate Authority (CA), which establishes end-to-end trust with a hierarchy of CAs, because the researchers take a similar approach to determine a domain is benign *if it is owned*

by a trusted owner or vouched by a trusted source. Yet, vouch is very different from certificate in that 1) it is not cryptographically signed and it can be forged, 2) the vouch is often unstructured text and sometimes ambiguous, 3) there is no agreement about which sites can be trustworthy vouchers. Our transitive trust model addresses the issues described above with a clean definition of IOB and an iterative search-and-check schema to find trusted IOBs from Internet. In the remainder of this section, we describe the design details of the transitive trust model and its implementation in `IOBHunter`.

B. Transitive Trust Model of IOB

In Figure 5, we illustrate the diagram of the transitive trust model to find trusted IOBs. Given a detected FQDN, `IOBHunter` first obtains the search engine results for the FQDN (which is termed as search target). Then, for each URL in the search results, we check if its indexed page contains an IOB of the search targets. The URL containing IOB is called an *IOB source*, and we verify if the trust of the IOB source can be transited to search targets under the *trust transit policy*. If the trust transition is valid, we further check if the IOB source is the *root of trust* and claim that trusted IOBs are found if true. Otherwise, a search-and-check process is conducted iteratively until the root of trust is reached or no new search result is returned. To avoid indefinite loops, a limit of search iterations is set. Compared to the evidence search described in §III-B6, the search-and-check process of `IOBHunter` mitigates the issue of IOB forging with the proposed trust model.

As we can see, such a transitive trust model involves three main problems: the definition of an IOB, the root of trust, and the trust transit policy. Next, we describe how they are solved.

1) Formal definition of an IOB: We argue that IOB has to be carefully defined to avoid mistrust or misclassification on a domain. Usually, a lot of text description about a domain can be found through search engines, but not all of them can be considered as IOB. For example, a technical blog by security researchers might mention a malicious domain that is exploited for a cyber-attack. Such Web content actually serves as an IOC instead of an IOB. Sometimes, the text description about a domain is neither an IOB nor an IOC. For instance, `radar.cloudflare.com` provides the basic information of a domain, including passive DNS, SSL certificate, and Whois data. However, they do not determine if the domain is benign or malicious. Finally, with respect to websites that describe domains, they can be created by an attacker or abused to forge trust. For instance, attackers can set up a homepage on social network apps such as Facebook and Instagram that point to malicious domains and claim its trustworthy.

Obviously, the three aforementioned cases do not have IOBs, and we avoid them by restricting the IOB to be a piece of Web content that *attributes* a domain to a *legitimate app or organization*. If this piece of Web content comes from a trusted source, it is called a *trusted IOB*. As such, the first two cases (malicious domains described in a technical blog and domains listed in informational websites) do not have IOBs. The third case (social network app vouching a domain) has IOBs but not trusted IOBs, considering that the sources can be manipulated.

2) Root of trust: Similar as the CA chain-of-trust model, we also need to define a root of trust, which has the highest level of trust to vouch other sources. In our model, we consider the *websites owned by government* to be the root of trust. For example, in an ideal case, if government documents or websites mention that a detected domain is used for legitimate purposes, this IOB will be considered trustworthy. We are aware that our root of trust is quite restrictive (only government sites), and we choose government sites because the published content is expected to be carefully verified by officials. Although government websites might be compromised, the attackers in this case prefer website defacement [79], and describing their malicious domains on government sites would more likely expose the malicious domains.

Unfortunately, for most domains detected in practice, no IOB for them could be found directly from government sites. Instead, IOBs are usually found *indirectly*, where an IOB source is certified by government organizations and this source contains IOBs for a detected domain. It might require multiple website visits to locate the description of the target FQDN from the root of trust. Moreover, in some cases, IOBs can be found only for root domains of detected domains. Therefore, we use the trust transit policy to define how trust can be transited from the root of trust to the target FQDNs and between root domains and their subdomains.

3) Trust transit policy: We define three policies. Our general rule is that if two entities (*i.e.,* root domains, subdomains, FQDN, and URL/web content) share *the same owner*, their trust can be transited. The Web-content→Search-Target policy is established given the IOB definition.

Root-domain↔Subdomain. This type of trust transit applies to cases where an IOB is found for the root domain of a subdomain and vice versa. Usually, domains under the same root domain are managed by the same owner, and thus trust can be transited. However, cases exist that a subdomain and its root domain are managed by different owners. For instance, CDN, dynamic DNS, *etc.* allow arbitrary public users to obtain subdomains under their root domains. Therefore, if a subdomain of a root domain is found to be controllable by another party, the trust cannot be transited.

FQDN→URL/Web-content. Since we define IOB to be a piece of Web content and the trustworthiness of the Web content needs to be certified, we allow the Web content to inherit the trust from its hosting FQDN. Similarly, a URL can inherit the trust from the FQDN where it is hosted. Like Root-domain↔Subdomain, there are two scenarios in which the trust of an FQDN cannot be transited to its URLs or the Web content: URLs generated for public users (*e.g.,* online storage and data sharing services), and the website that allows arbitrary public users to publish content (*e.g.,* online forums, social networks).

Web-content→Search-Target. Finally, the trust of Web content can be transited to the search target if the content describing the search target satisfies IOB definition in §IV-B1.

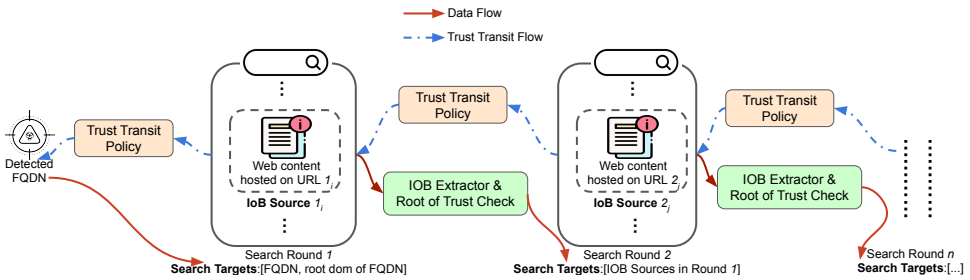

Fig. 5: Diagram of the transitive trust model for IOB.

4) A Running Example: We present an example for a real FP case that can be corrected with our transitive trust model, which is shown in Appendix §C.

C. Implementation of `IOBHunter`

Implementing `IOBHunter` needs to address two major challenges. First, IOBs from Web content are often unstructured. Second, domain ownership and the intention of the description (*e.g.,* IOC or IOB) cannot be easily determined. We found Large Language Models (LLMs) like GPT-4o have a unique advantage in addressing the issues of text parsing and knowledge extraction (reviewed in §VII), so we build `IOBHunter` on top of LLM. To further ensure the factuality of the responses from LLMs, we leverage chain-of-thought (CoT) [76] when designing prompts, which has shown its effectiveness in detecting online hate texts [75] and memes [83] with high accuracy. `IOBHunter` consists of multiple phases that use data obtained in multiple search rounds, and we show its pseudo-code in Appendix §E.

Phase 1: Search targets. Before each search round, we first need to decide the search targets. `IOBHunter` checks whether the detected FQDN and FQDNs in the IOB sources found in the previous round are root domains. If so, they are added as targets for the next search round. Otherwise, we assess whether the FQDNs are managed by the same owner as their root domains using a proprietary database compiled by ***SV***. If so, both the FQDNs and their root domains are added as search targets.

Phase 2: Search result retrieval. Once search targets are determined, `IOBHunter` obtains the search results using the same approach as ***SV*** security researchers in §III-B6.

Phase 3: IOB extraction. For each search result from Phase 2, `IOBHunter` performs multiple checks using the prompt presented in Appendix §D. First, `IOBHunter` checks whether the searched content contains the corresponding search target (Step 1 Appendix §D). Second, `IOBHunter` checks the searched content to decide whether the target FQDN or its root domain is associated with a legitimate application or is owned by an organization. Next, the system evaluates whether the content in the search result is published by arbitrary public users or originates from a content hosting or aggregation platform, *i.e.,* checking against trust transit policies. If both checks are true, the process continues to check whether the IOB source is an official government website (*e.g.,* ending in `.gov`). If so, a trusted IOB is found for the detected FQDN. Otherwise, the trustworthiness of an IOB source is undecided (*i.e.,* `tbd` in Step 5), and `IOBHunter` repeats the process in a new search round from Phase 1 for the IOB source.

Phase 4: Trust propagation. Since trust transit policies have been checked in Phase 3, it is straightforward to combine the results from all search rounds to get the final results. Starting from the root of trust in the search round $i+1$, we propagate its trust to IOB sources whose trustworthiness is undecided in the search round i. This trust propagation is performed recursively until the detected FQDN is reached. After trust propagation, if one IOB source in search round 2 (*i.e.,* the initial IOB source) is trusted, a trusted IOB is found for the detected FQDN.

Phase 5: Termination conditions. `IOBHunter` terminates when sufficient trusted IOBs are found and when all IOB sources in a search round are untrusted (*i.e.,* no trusted IOB found). In practice, a single trusted IOB is sufficient for FP mitigation, and thus it is our default configuration. We also set a limit for search iterations (3 by default).

Optimization. To optimize cost and throughput, a cache is maintained for the IOB sources that have been assessed. Both negative and positive results can be cached so that an IOB source does not need reassessment if it has been verified before. This idea resembles Retrieval Augmented Generation (RAG) in LLM domain [47] to augment a prompt with additional knowledge from the LLM user.

D. Separating Compromised Domains From Real FPs

As learned from **Lesson 4**, many FP CRs are for compromised websites. Since these websites are benign and legitimate in essence, `IOBHunter` could find trusted IOBs for them. However, in practice, these compromised websites should not be classified as FPs. Instead, we need to separate compromised websites from real FPs so that `IOBHunter` can be automated in production. To achieve this, we adopt a simple policy that given a domain with trusted IOBs, if solid malicious evidence is also present, the domain is classified as compromised; otherwise, it is an FP. This policy is also used in our evaluation.

The main challenge to implement this policy lies in the fact that the definition of solid malicious evidence is an open problem and could differ greatly between security vendors. The perfect implementation of this policy is not the focus of this work. Instead, we currently simply follow the same imple-

mentation as adopted by ***SV*** which has performed quite well in production. Specifically, solid malicious evidence includes i) malicious activities confirmed in security blogs and technical articles from a list of manually curated trusted sources, and ii) websites contain malicious content that exhibits malicious activities during dynamic analysis or is flagged by more than five VirusTotal vendors.

V. Evaluation

In this section, we assess the efficacy of `IOBHunter` and aim to answer three main questions:

- How accurate is `IOBHunter`?
- Is `IOBHunter` efficient and economical?
- Can `IOBHunter` complement popularity-based allowlists to mitigate FPs in production?

To answer these questions, we evaluate `IOBHunter` in two setups. First, the FP CR dataset from our measurement study is used as ground truth, which facilitates the evaluation of accuracy and efficiency. Since the FP CR dataset is biased towards true FPs, we further evaluate `IOBHunter` in a real-world setup and apply `IOBHunter` to *all* malicious domains detected by ***SV*** from 2025-03-01 to 2025-05-01.

A. Evaluation With FP CRs

We first evaluate `IOBHunter` using the FP CR dataset described in §III. We use GPT-4o [18] as the default LLM with the following parameters: temperature = 0.1, top_p = 0.95, frequency_penalty = 0, and presence_penalty = 0. We also assess the impact of LLMs using two other models: Qwen3-32B [19] and Gemini-2.5-Flash [20]. Qwen3-32B is open-weights and running it locally can address the privacy concerns of sending data to remote LLM vendors.

Evaluation metrics. For effectiveness, we use recall and precision as the primary metrics. However, although we have the ground truth about FPs, we do not have the ground truth about which FPs have trusted IOBs. Since it is impractical to manually verify the trusted IOBs for all FP CRs, we use accepted and rejected FPs with potential benign indicators on the Internet (*i.e.,* 66,639 accepted CRs and 669 rejected CRs with the first type of benign evidence in §III-B6) to *approximate* the ground truth. Specifically, for an accepted FP, if a trusted IOB is found, it is considered a true detection (*i.e.,* True IOB), otherwise a false negative (*i.e.,* Missing IOB). Similarly, for a rejected FP, if a trusted IOB is found, it is considered a false detection (*i.e.,* False IOB), otherwise a true negative (*i.e.,* No IOB). Note that, to simplify the presentation, trusted IOBs for compromised websites are counted as False IOB. We define the number of CRs with True IOB, Missing IOB, False IOB and No IOB as CR_T, CR_M, CR_F and CR_N. Therefore, precision and recall are defined as: Precision $= \frac{CR_T}{CR_T+CR_F}$; Recall $= \frac{CR_T}{CR_T+CR_M}$.

Evaluation results. Table VII shows the results of `IOBHunter` using three different LLM models. We can see that the three models perform equally well, with a precision greater than 99% and a recall around 70%.

	`IOBHunter`		
	GPT-4o	**Gemini2.5-Flash**	**Qwen3-32B**
CR w/ True IOB	45,591	47,082	45,961
CR w/ Missing IOB	21,048	19,557	20,678
CR w/ False IOB	359	453	439
CR w/ No IOB	310	216	230
Precision	99.22%	99.05%	99.04%
Recall	68.41%	70.65%	68.97%

TABLE VII: Evaluation results of `IOBHunter` on CR dataset.

We further dive into cases with missing IOBs and false IOBs. We present only the results for GPT-4o here, and the results for the other two models share similar characteristics and conclusions. First, for the 21,048 cases with missing IOBs, we sampled 100 cases for manual verification. While all of them are correctly classified by `IOBHunter`, we also notice a pattern. For 13,443 out of 21,048 cases, there is benign evidence on social media platforms such as `reddit.com` and `instagram.com`. Manual investigation shows that they are valid IOBs. However, by design, `IOBHunter` does not identify them as trusted. We further discuss this issue in §VI. In addition, the top five detectors that detected the cases with missing IOBs are malicious web content detector (15,628), third-party feeds (6,315), domain resolution pattern based detector (911), malicious newly-registered domain detector (298), and dictionary DGA detector(192). Note that, one case can be detected by multiple different detectors and we count them to each detector.

Among the 359 cases with false IOBs, 336 are actually compromised according to the policy in §IV-D. Since there were still malicious activities when the FP CRs were submitted, the CRs were rejected by ***SV*** researchers. Interestingly, no subsequent FP CRs are submitted for these rejected cases. For the other 23 cases, 17 are subdomains of security services and pentesting tools that are used in both benign and malicious scenarios. Instead of simply classifying them as false detections of `IOBHunter`, we argue that they should be taken care of specially in practice. Finally, the remaining five cases are potential false detections of `IOBHunter`. In summary, after excluding compromised cases and security tools, there are only a few false IOBs by `IOBHunter`.

Cost of `IOBHunter`. Since millions of malicious domains are detected every month, the cost of `IOBHunter` is an important factor for real-world deployment. Table VIII shows the cost of three LLM models. Considering that all three models perform equally well, `IOBHunter` could be as cheap as $74 per million FQDNs, which is affordable for security service providers. Another main source of cost comes from Gemini Grounding which is priced at $35 per 1000 requests. The cost will be prominent if a large number of domains are searched, and we are exploring a more economical solution such as Google Programmable Search Engine [17].

Throughput of `IOBHunter`. Table IX shows the statistical latency of `IOBHunter`. The latency of `IOBHunter` consists of two parts, searching with Gemini Grounding and prompts to analyze search results. It takes 35.5 seconds on average for

Model	Input Cost	Output Cost	Actual Cost Per FQDN
GPT-4o	$2.50	$10.00	$0.0023
Gemini-2.5-Flash	$0.15	$0.60	$0.00014
Qwen-3-32B	$0.10	$0.30	$0.000074

TABLE VIII: Token based pricing of LLMs per 1M input and output tokens. The last column shows the actual average cost per FQDN on our FP CR dataset.

	Min	Max	Avg	Med
Search w/ Gemini Grounding	4.2s	322.7s	35.5s	21.4s
`IOBHunter` (gpt-4o)	0.8s	20.2s	1.4s	1.4s
`IOBHunter` (gemini2.5-flash)	0.4s	4.2s	0.8s	0.8s
`IOBHunter` (qwen3-32B)	0.5s	1065.4s	38.4s	19.5s

TABLE IX: Latency of prompts on FP CR dataset.

one Gemini Grounding search. The prompts to analyze search results are much more efficient, around 1 second for GPT-4o and Gemini2.5-flash. For Qwen3-32B, the latency is much higher because our evaluation uses OpenRouter API [19]. Hosting Qwen3-32B locally with a powerful GPU is likely to yield better performance. In summary, the bottleneck in the current implementation of `IOBHunter` is the search with Gemini Grounding, and we expect it to be greatly improved after switching to Google Programmable Search Engine [17].

B. Real-World Evaluation

In §V-A, we evaluate `IOBHunter` using our FP CR dataset. However, this dataset is biased towards true FPs. In particular, it remains unknown whether `IOBHunter` also finds trusted IOBs for true malicious domains and leads to false negatives of detectors. Therefore, to demonstrate the value of `IOBHunter` in a more realistic and practical setting, we deploy `IOBHunter` in production, as shown in Figure 6. `IOBHunter` runs after the allowlists are checked and is triggered for each malicious FQDN detected in ***SV*** (both in-house detectors and third-party feeds) from 2025-03-01 to 2025-05-01. `IOBHunter` runs in the shadow mode in which the results of `IOBHunter` do not prevent detected domains from being flagged as malicious. The primary metric for this evaluation is the number of FPs confirmed in production. To comply with ***SV***'s policy, only GPT-4o is used in this deployment. Since all three models perform comparably as shown in the previous evaluation, we believe that the result from GPT-4o is representative.

During this period, ***SV*** detects 15,106,720 unique FQDNs whose root domains are registered. Checking them all is too costly for `IOBHunter`. As such, we reduce the set by excluding FQDNs that are unlikely to be FPs. Specifically, researchers in ***SV*** have extracted patterns of malicious domains involved in hundreds of known campaigns. Domains that match these patterns are unlikely to be FPs and therefore do not need to be checked by `IOBHunter`. After filtering, 15M FQDNs are reduced to 1,065,587 under 654,987 root domains. Note that although tens of millions DGA domains are detected every day (as shown in Table V), almost all of them are nonexistent domains (*i.e.,* whose root domains are not

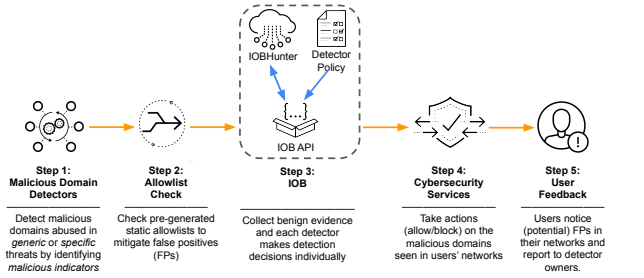

Fig. 6: `IOBHunter` is deployed after existing allowlists. `IOBHunter` runs in the shadow mode where we only log the results for manual evaluation but do not automatically affect the verdicts of detected domains.

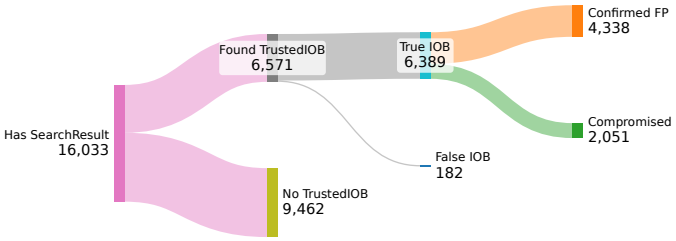

Fig. 7: Evaluation results of a real-world deployment of `IOBHunter` from 2025-03-01 to 2025-05-01.

registered by anyone), which would not cause actual damage to users. We remove all such cases.

For 16,033 out of the 1,065,587 FQDNs, `IOBHunter` could find valid search results from search engines. ~98.5% FQDNs yielded no valid results simply because nothing can be found from Google. This is common for malicious domains and domains without Web content. For example, knotnormal-productions[.]online is detected as malicious. But, no single result about this domain can be found from Google Search. Figure 7 shows the evaluation results of the 16,033 cases. `IOBHunter` finds trusted IOBs for 6,571 cases. After manually investigating these cases and confirm with the researchers in ***SV***, we find that 4,338 cases are confirmed to be FPs. The verdicts of these 4,338 cases have been corrected from malicious to benign by ***SV*** researchers based on the results of `IOBHunter`. In addition, according to §IV-D, 2,051 cases are classified as compromised. This demonstrates that `IOBHunter` can also effectively distinguish compromised domains from those owned by attackers, which can potentially address user complaints (§III-B5). These two types of results are classified as true IOBs by `IOBHunter` (6,389 in total). On the other hand, there are 182 potential false IOBs by `IOBHunter`. They are mainly caused by LLM hallucinations [81]. For example, in 15 out of 182 cases, LLM claims that IOBs are found from sources such as `scamadviser.com`, but the relevant IOB description does not exist after we inspect the sources. Finally, for the 9,462 cases where `IOBHunter` does not find trusted IOBs, we sampled 100 cases and manually confirm that the result is correct.

	Confirmed FP	Compromised	False IOB	No IOB
IOBHunter	4,338	2,051	182	9,462
Tranco-10K	0	0	0	1
Tranco-100K	0	0	0	13
Tranco-500K	0	2	0	38
Tranco-1M	6	8	0	100

TABLE X: Cross-check of IOBHunter results with Tranco-2025-05-01.

C. Comparison with Tranco

Finally, to demonstrate the value of IOBHunter in mitigating FPs, we compare IOBHunter with the Tranco list. In particular, for the 16,033 cases that have search results, we check how many of them appear in the Tranco list. Table X shows the results for the Tranco list on 2025-05-01. The results on other dates during 2025-03-01 and 2025-05-01 are similar. We can see that only 6 of 4,338 (0.1%) confirmed FPs and 8 of 2,051 (0.4%) compromised cases could potentially be mitigated by Tranco. Note that although the Tranco list is used by ***SV***, these 14 cases were not caught because ***SV*** slightly customized it and some detectors can bypass the Tranco list as described in §II-D. In addition, 100 of 9,462 (1%) cases without trusted IOB appear in the Tranco list. Manual investigation confirms that 14 are truly malicious (*i.e.,* not FPs), and for the rest, we cannot definitively decide whether they are true FPs or not. These results demonstrate the unique value of IOBHunter in reducing FPs in practice.

VI. Discussion

Limitations and future work. In §III-C, we have discussed the threat to validity of the measurement study. Here, we discuss the potential limitations of IOBHunter. First, IOBHunter can introduce false negatives for malicious domain detectors if trusted IOBs were identified on the malicious domains. To address this issue, we configure IOBHunter to run in shadow mode as shown in Figure 6 and its contribution to the final verdict can be tuned. Second, the recall of IOBHunter can be further improved, and we reveal that the main reason in §V-A is that IOBs from social media platforms are ignored by IOBHunter, determined by the transitive trust model. A potential solution could be to profile the trustworthiness of social media users and transit the trust to reputable users. Third, the problem of residual trust is relevant in our setting. For instance, a domain expires and then gets registered by another user. If IOBs of the domain are not invalidated, the new owner can exploit the residual trust and evade detection by security services. Actively monitoring the time of registration updates, which is not anonymized in Whois data, could address this issue.

We implement the transitive trust model to find trusted IOBs. Compared to the CA chain-of-trust model, our policies are relatively loose. For example, we allow subdomains to inherit trust from their root domain, which is not allowed by default by CA unless a wildcard certificate is used [64]. We argue that strict policies like CA's would drastically reduce the number of trusted IOBs that can be found, and we leverage LLMs to analyze IOBs and validate trust transition to address the potential issues of relaxed policies.

More use cases. In this paper, we have demonstrated that IOBHunter can be used to mitigate FPs and distinguish between malicious and compromised websites. We expect IOBHunter to be applied in more scenarios. First, for security vendors, IOBHunter can facilitate manual investigation of FP CRs (*i.e.,* moving Step 3 to be after Step 5 in Figure 6). Second, for Security Operations Centers (SOC) and security analysts, IOBHunter can provide more context as threat intelligence which can help investigate security alerts and incidents. Finally, some indicators collected by IOBHunter could potentially be used as features for machine learning models of detectors. In some of these use cases, even untrusted IOBs are also valuable.

VII. Related Work

We have covered malicious domain detection and existing common practices to avoid FP in §II. In this section, we discuss other related work.

Cyber threat intelligence (CTI) gathering and parsing. Several previous works focus on the extraction of IOC from unstructured CTI reports. iACE extracts a set of putative IOC tokens and their contexts from technical articles and then uses graph mining to generate IOC items [50]. Extractor [65] and TTPDrill [41] use Natural Language Processing (NLP) to extract attack behaviors and threat actions from CTI reports. Tweezers implements an event attribution-centric tweet embedding method to gather security events from Twitter [30]. Since the emergence of powerful LLMs, researchers have investigated the feasibility of using LLMs for CTI. LocalIntel leverages LLMs to generate and contextualize threat intelligence by combining global threat reports and organization-specific knowledge [59]. aCTIon uses LLMs to automate structured CTI extraction from verbose articles [69]. CTIKG builds a security-oriented knowledge graph from CTI articles based on LLMs [40]. CTINexus designs an automatic prompt construction strategy and leverages optimized in-context learning of LLMs to construct CTI knowledge graphs [28]. As a result, CTINexus can adapt to various ontologies with minimal annotated examples. In contrast to these works, IOBHunter seeks to accomplish the *opposite* goal by identifying the trusted IOB of domains on the Internet, a task that presents unique technical challenges. To our knowledge, IOBHunter is the first system that leverages LLMs for automated IOB collection from unstructured public information.

LLM for web content analysis. Mind2Web [31], WebGUM [34] and Gur *et al.* [36] [35] employ LLMs to complete complex tasks, *e.g.,* form filling, on websites following human's high-level instructions. To achieve task automation on web, these systems analyze HTML documents and predict the elements for interaction and the corresponding operations. Such systems are generalist agents for web and they might improve IOBHunter in crawling websites and searching IOBs. Recently, some works employed LLMs to detect toxic content, online hate and phishing [38] [75] [74] [55]. He *et al.* explored

prompt learning for toxic content detection and yielded an improvement of 10% compared to baselines [38]. Vishwamitra *et al.* leveraged the chain of thought to identify online hate and showcased 10.59% to 88% improvement in accuracy [75]. Thomas *et al.* explored the use of LLMs to enhance the expertise of human raters and achieved a 9-11% increase in precision and recall [74]. Liu *et al.* utilized LLMs to infer the intention of the brand from logo descriptions on websites [55]. The task performed by `IOBHunter` is fundamentally different, which identifies trusted benign evidence of domains from Web content.

Compromised domain detection. Our work is also related to systems that aim to distinguish compromised benign domains from attacker-owned domains. Existing systems [68] [58] mainly use lexical features, domain popularity, Whois, VirusTotal, passive DNS, and TLS certificate to classify compromised and maliciously registered domains. `IOBHunter` represents a totally different approach based on IOB collected from the Internet, which can complement existing systems.

VIII. Conclusion

This paper has conducted the first large-scale measurement study of FPs in production. Our study is built on a 6-year FP dataset reported by users of a leading security vendor. We reveal that FPs are still prevalent in production, and existing popularity-based top domain lists cannot effectively address them. Another key finding is that indicators of benignity (IOBs) for ∼55% FPs can be found on the Internet. Motivated by our findings, we make the first effort to automatically identify IOBs on the Internet. To this end, we propose a *transitive trust model* for IOB and implement it in a system called `IOBHunter` that leverages LLM and chain-of-thought (CoT). Our evaluation on a labeled dataset shows that `IOBHunter` can achieve 99.22% precision and 68.6% recall. Finally, a two-month deployment of `IOBHunter` in real-world has identified 4,338 confirmed FPs and 2,051 compromised domains.

IX. Ethics Considerations

Our research follows the policies of the legal department of ***SV***. Our dataset does not contain personally identifiable information (PII), *e.g.,* users' names and their IP addresses. We avoid connecting CRs with their submitters, hence there are no measurement results by submitters' regions, demographics, *etc.* The comments submitted by users in CRs are only analyzed by researchers in ***SV*** on secure internal servers, and the data has not been shared with any outside party.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful and constructive comments. The authors would also like to thank all researchers who implemented the detectors and those who conducted the CR reviews at Palo Alto Networks. Any opinions, findings, and conclusions expressed in this material are those of the author(s) and do not necessarily reflect the views of Palo Alto Networks or the sponsors.

References

[1] https://blog.cloudflare.com/threat-detection-machine-learning-models/. Accessed: 2025-05-07.
[2] https://www.akamai.com/blog/security-research/newly-observed-domains-discovered-13-million-malicious-domains#:~:text=Akamai%20researchers%20have%20flagged%20almost,(NODs)%20that%20successfully%20resolved. Accessed: 2025-05-07.
[3] https://community.akamai.com/community/cloud-security/blog/2014/11/06/the-ugly-truth-behind-the-practice-of-ip-whitelisting. Accessed: 2024-12-05.
[4] https://datatracker.ietf.org/doc/html/rfc5782. Accessed: 2024-12-05.
[5] https://en.wikipedia.org/wiki/Alexa_Internet. Accessed: 2024-12-05.
[6] https://arstechnica.com/information-technology/2017/11/new-quad9-dns-service-blocks-malicious-domains-for-everyone/. Accessed: 2024-12-05.
[7] https://www.netresec.com/?page=Blog&month=2013-10&post=DNS-whitelisting-in-NetworkMiner. Accessed: 2024-12-05.
[8] https://tranco-list.eu/. Accessed: 2025-05-26.
[9] https://majestic.com/reports/majestic-million/. Accessed: 2025-02-26.
[10] https://umbrella.cisco.com/blog/cisco-umbrella-1-million. Accessed: 2025-02-26.
[11] https://web.archive.org/web/20200105223115/https://www.quantcast.com/top-sites/. Accessed: 2025-02-26.
[12] https://developer.chrome.com/docs/crux/methodology/. Accessed: 2025-02-26.
[13] https://otx.alienvault.com/. Accessed: 2025-02-19.
[14] https://www.microsoft.com/en-us/bing/apis/bing-web-search-api. Accessed: 2025-01-27.
[15] https://azure.microsoft.com/en-us/updates?id=483570. Accessed: 2025-05-27.
[16] https://ai.google.dev/gemini-api/docs/grounding. Accessed: 2025-01-27.
[17] https://developers.google.com/custom-search/. Accessed: 2025-05-07.
[18] https://azure.microsoft.com/en-us/pricing/details/cognitive-services/openai-service/. Accessed: 2025-05-27.
[19] https://openrouter.ai/qwen/qwen3-32b. Accessed: 2025-06-02.
[20] https://deepmind.google/models/gemini/flash. Accessed: 2025-06-02.
[21] Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. Building a dynamic reputation system for dns. In *USENIX Security*, 2010.
[22] Manos Antonakakis, Roberto Perdisci, Wenke Lee, Nikolaos Vasiloglou II, and David Dagon. Detecting malware domains at the upper DNS hierarchy. In *USENIX Security*, 2011.
[23] Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. From throw-away traffic to bots: detecting the rise of dga-based malware. In *USENIX Security Symposium*, 2012.
[24] Jan Bayer, Sourena Maroofi, Olivier Hureau, Andrzej Duda, and Maciej Korczynski. Building a resilient domain whitelist to enhance phishing blacklist accuracy. In *APWG eCrime*, 2023.
[25] Leyla Bilge, Engin Kirda, Christopher Krügel, and Marco Balduzzi. Exposure: Finding malicious domains using passive dns analysis. In *NDSS*, 2011.
[26] Renée Burton and Laura Rocha. Whitelists that work: Creating defensible dynamic whitelists with statistical learning. In *APWG Symposium on Electronic Crime Research (eCrime)*, 2019.
[27] Zhouhan Chen and Juliana Freire. Discovering and measuring malicious url redirection campaigns from fake news domains. In *2021 IEEE Security and Privacy Workshops (SPW)*, 2021.
[28] Yutong Cheng, Osama Bajaber, Saimon Amanuel Tsegai, Dawn Song, and Peng Gao. Ctinexus: Leveraging optimized llm in-context learning for constructing cybersecurity knowledge graphs under data scarcity. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2025.
[29] Corinna Cortes and Vladimir Naumovich Vapnik. Support-vector networks. *Machine Learning*, 2004.
[30] Jian Cui, Hanna Kim, Eugene Jang, Dayeon Yim, Kicheol Kim, Yongjae Lee, Jin-Woo Chung, Seungwon Shin, and Xiaojing Liao. Tweezers: A framework for security event detection via event attribution-centric tweet embedding. In *NDSS*, 2025.
[31] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[32] Kun Du, Hao Yang, Zhou Li, Haixin Duan, Shuang Hao, Baojun Liu, Yuxiao Ye, Mingxuan Liu, XiaoDong Su, Guang Liu, Zhifeng Geng, Zaifeng Zhang, and Jinjin Liang. Tl;dr hazard: A comprehensive study of levelsquatting scams. In *SecureComm*, 2019.

[33] David Erickson, Martín Casado, and Nick McKeown. The effectiveness of whitelisting: a user-study. In *International Conference on Email and Anti-Spam*, 2008.

[34] Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust, Shixiang Shane Gu, and Izzeddin Gur. Multimodal web navigation with instruction-finetuned foundation models. In *ICLR*, 2024.

[35] Izzeddin Gur, Hiroki Furuta, Austin V Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. In *ICLR*, 2024.

[36] Izzeddin Gur, Ofir Nachum, Yingjie Miao, Mustafa Safdari, Austin V Huang, Aakanksha Chowdhery, Sharan Narang, Noah Fiedel, and Aleksandra Faust. Understanding HTML with large language models. In *EMNLP*, 2023.

[37] Shuang Hao, Alex Kantchelian, Brad Miller, Vern Paxson, and Nick Feamster. Predator: Proactive recognition and elimination of domain abuse at time-of-registration. In *CCS*, 2016.

[38] Xinlei He, Savvas Zannettou, Yun Shen, and Yang Zhang. You Only Prompt Once: On the Capabilities of Prompt Learning on Large Language Models to Tackle Toxic Content . In *IEEE S&P*, 2024.

[39] Thorsten Holz, Christian Gorecki, Konrad Rieck, and Felix C. Freiling. Measuring and detecting fast-flux service networks. In *Network and Distributed System Security Symposium (NDSS)*, 2008.

[40] Liangyi Huang and Xusheng Xiao. CTIKG: LLM-powered knowledge graph construction from cyber threat intelligence. In *First Conference on Language Modeling*, 2024.

[41] Ghaith Husari, Ehab Al-Shaer, Mohiuddin Ahmed, Bill Chu, and Xi Niu. Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources. In *ACSAC*, 2017.

[42] Collin Jackson, A. Barth, A. Bortz, Weidong Shao, and Dan Boneh. Protecting browsers from dns rebinding attacks. In *ACM CCS*, 2007.

[43] Issa Khalil, Ting Yu, and Bei Guan. Discovering malicious domains through passive dns data graph analysis. In *AsiaCCS*, 2016.

[44] Kyungtae Kim, I Luk Kim, Chung Hwan Kim, Yonghwi Kwon, Yunhui Zheng, Xiangyu Zhang, and Dongyan Xu. J-force: Forced execution on javascript. In *WWW*, 2017.

[45] Panagiotis Kintis, Najmeh Miramirkhani, Charles Lever, Yizheng Chen, Rosa Romero-Gómez, Nikolaos Pitropakis, Nick Nikiforakis, and Manos Antonakakis. Hiding in plain sight: A longitudinal study of combosquatting abuse. In *ACM CCS*, 2017.

[46] Clemens Kolbitsch, Benjamin Livshits, Benjamin Zorn, and Christian Seifert. Rozzle: De-cloaking internet malware. In *IEEE S&P*, 2012.

[47] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[48] Zhou Li, Sumayah Alrwais, XiaoFeng Wang, and Eihal Alowaisheq. Hunting the red fox online: Understanding and detection of mass redirect-script injections. In *IEEE S&P*, 2014.

[49] Zhou Li, Sumayah Alrwais, Yinglian Xie, Fang Yu, and XiaoFeng Wang. Finding the linchpins of the dark web: a study on topologically dedicated hosts on malicious web infrastructures. In *IEEE S&P*, 2013.

[50] Xiaojing Liao, Kan Yuan, XiaoFeng Wang, Zhou Li, Luyi Xing, and Raheem Beyah. Acing the ioc game: Toward automatic discovery and analysis of open-source cyber threat intelligence. In *CCS*, 2016.

[51] Yun Lin, Ruofan Liu, Dinil Mon Divakaran, Jun Yang Ng, Qing Zhou Chan, Yiwen Lu, Yuxuan Si, Fan Zhang, and Jin Song Dong. Phishpedia: A hybrid deep learning based approach to visually identify phishing webpages. In *30th USENIX Security Symposium*, 2021.

[52] Daiping Liu, Ruian Duan, and Jun Wang. Resolution Without Dissent: In-Path Per-Query Sanitization to Defeat Surreptitious Communication Over DNS . In *IEEE Symposium on Security and Privacy (SP)*, 2025.

[53] 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 *CCS*, 2017.

[54] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *IEEE International Conference on Data Mining (ICDM)*, 2008.

[55] Ruofan Liu, Yun Lin, Xiwen Teoh, Gongshen Liu, Zhiyong Huang, and Jin Song Dong. Less defined knowledge and more true alarms: Reference-based phishing detection without a pre-defined reference list. In *33rd USENIX Security Symposium*, 2024.

[56] Chaoyi Lu, Baojun Liu, Yiming Zhang, Zhou Li, Fenglu Zhang, Haixin Duan, Ying Liu, Joann Qiongna Chen, Jinjin Liang, Zaifeng Zhang, Shuang Hao, and Min Yang. From whois to whowas: A large-scale measurement study of domain registration privacy under the gdpr. *NDSS*, 2021.

[57] Pratyusa K. Manadhata, Sandeep Yadav, Prasad Rao, and William Horne. Detecting malicious domains via graph inference. In *ESORICS*, 2014.

[58] Sourena Maroofi, Maciej Korczyński, Cristian Hesselman, Benoît Ampeau, and Andrzej Duda. Comar: Classification of compromised versus maliciously registered domains. In *IEEE EuroS&P*, 2020.

[59] Shaswata Mitra, Subash Neupane, Trisha Chakraborty, Sudip Mittal, Aritran Piplai, Manas Gaur, and Shahram Rahimi. Localintel: Generating organizational threat intelligence from global and local cyber knowledge. *arXiv preprint arXiv:2401.10036*, 2024.

[60] Asaf Nadler, Avi Aminov, and Asaf Shabtai. Detection of malicious and low throughput data exfiltration over the dns protocol. *Computers & Security*, 2019.

[61] Vern Paxson, Mihai Christodorescu, Mobin Javed, Josyula Rao, Reiner Sailer, Douglas Lee Schales, Mark Stoecklin, Kurt Thomas, Wietse Venema, and Nicholas Weaver. Practical comprehensive bounds on surreptitious communication over DNS. In *USENIX Security*, 2013.

[62] Victor Le Pochat, Tom van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. *NDSS*, 2018.

[63] Elsa Rodríguez, Radu Anghel, Simon Parkin, Michel van Eeten, and Carlos Gañán. Two sides of the shield: Understanding protective DNS adoption factors. In *USENIX Security*, 2023.

[64] P Saint-Andre and J Hodges. Representation and verification of domain-based application service identity within internet public key infrastructure using x. 509 (pkix) certificates in the context of transport layer security (tls). Technical report, 2011.

[65] K. Satvat, R. Gjomemo, and V.N. Venkatakrishnan. Extractor: Extracting Attack Behavior from Threat Reports . In *IEEE EuroS&P*, 2021.

[66] Samuel Schüppen, Dominik Teubert, Patrick Herrmann, and Ulrike Meyer. Fanci: feature-based automated nxdomain classification and intelligence. In *USENIX Security Symposium*, 2018.

[67] Silvia Sebastián, Raluca-Georgia Diugan, Juan Caballero, Iskander Sanchez-Rola, and Leyla Bilge. Domain and website attribution beyond whois. In *ACSAC*, 2023.

[68] 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*, 2021.

[69] Giuseppe Siracusano, Davide Sanvito, Roberto Gonzalez, Manikantan Srinivasan, Sivakaman Kamatchi, Wataru Takahashi, Masaru Kawakita, Takahiro Kakumaru, and Roberto Bifulco. Time for action: Automated analysis of cyber threat intelligence in the wild. *arXiv preprint arXiv:2307.10214*, 2023.

[70] Oleksii Starov, Yuchen Zhou, and Jun Wang. Detecting malicious campaigns in obfuscated javascript with scalable behavioral analysis. In *2019 IEEE Security and Privacy Workshops (SPW)*, 2019.

[71] Janos Szurdi, Balazs Kocso, Gabor Cseh, Jonathan Spring, Mark Felegyhazi, and Chris Kanich. The long "Taile" of typosquatting domain names. In *23rd USENIX Security Symposium*, 2014.

[72] Janos Szurdi, Meng Luo, Brian Kondracki, Nick Nikiforakis, and Nicolas Christin. Where are you taking me? understanding abusive traffic distribution systems. In *Proceedings of Web Conference*, 2021.

[73] N.I.S. Technology. *Guide to Application Whitelisting: NiST SP 800-167*. 2015.

[74] Kurt Thomas, Patrick Gage Kelley, David Tao, Sarah Meiklejohn, Owen Vallis, Shunwen Tan, Blaž Bratanič, Felipe Tiengo Ferreira, Vijay Kumar Eranti, and Elie Bursztein. Supporting Human Raters with the Detection of Harmful Content using Large Language Models . In *2025 IEEE Symposium on Security and Privacy (SP)*, 2025.

[75] Nishant Vishwamitra, Keyan Guo, Farhan Tajwar Romit, Isabelle Ondracek, Long Cheng, Ziming Zhao, and Hongxin Hu. Moderating New Waves of Online Hate with Chain-of-Thought Reasoning in Large Language Models . In *IEEE S&P*, 2024.

[76] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NIPS*, 2022.

[77] Colin Whittaker, Brian Ryner, and Marria Nazif. Large-scale automatic classification of phishing pages. In *NDSS '10*, 2010.
[78] Qinge Xie, Shujun Tang, Xiaofeng Zheng, Qingran Lin, Baojun Liu, Haixin Duan, and Frank Li. Building an open, robust, and stable Voting-Based domain top list. In *USENIX Security Symposium*, 2022.
[79] Ronghai Yang, Xianbo Wang, Cheng Chi, Dawei Wang, Jiawei He, Siming Pang, and Wing Cheong Lau. Scalable detection of promotional website defacements in black hat {SEO} campaigns. In *USENIX Security*, 2021.
[80] Penghui Zhang, Adam Oest, Haehyun Cho, Zhibo Sun, RC Johnson, Brad Wardman, Shaown Sarker, Alexandros Kapravelos, Tiffany Bao, Ruoyu Wang, Yan Shoshitaishvili, Adam Doupé, and Gail-Joon Ahn. Crawlphish: Large-scale analysis of client-side cloaking techniques in phishing. In *IEEE S&P*, 2021.
[81] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*, 2023.
[82] Shuofei Zhu, Jianjun Shi, Limin Yang, Boqin Qin, Ziyi Zhang, Linhai Song, and Gang Wang. Measuring and modeling the label dynamics of online Anti-Malware engines. In *USENIX Security*, 2020.
[83] Yong Zhuang, Keyan Guo, Juan Wang, Yiheng Jing, Xiaoyang Xu, Wenzhe Yi, Mengda Yang, Bo Zhao, and Hongxin Hu. I know what you meme! understanding and detecting harmful memes with multimodal large language models. *NDSS*, 2025.

APPENDIX A
DETAILS OF MALICIOUS DOMAIN DETECTION

Detection scope. Depending on what types of abused domains are targeted, detectors can be categorized into two types. First, a couple of detectors aim to identify the generic characteristics of malicious domains [21] [25] [22] [57] [49] [37] [43]. Another line of works instead focuses on specific types of domain abuses, such as domain squatting [71] [45] [32], domain generation algorithms (DGA) [23] [66], DNS rebinding [42], fast flux [39], domain shadowing [53], DNS tunneling [61] [60] [52] , and malicious websites [77] [51] [55] [46] [44] [70] [48].

Type of malicious indicators. To detect malicious domains, detectors rely on the characteristics and patterns of malicious domains (*i.e.,* malicious indicators). Malicious domain detectors generally rely on four types of indicators and many detectors combine multiple types of indicators. The first type of feature is derived from the lexical analysis of domain names, such as the name length and number of special characters. These features are the most commonly used [21] [25] [37] [71] [45] [32] [23] [66] [53] [61] [60] [52]. The rationale behind these features is that benign domains in general prefer meaningful words and mnemonic names, whereas malicious ones usually choose names with unique patterns, such as long random characters and look-alike names. Domain registration and Whois information, such as creation dates and domain registrants, are also widely used by many detectors [37] [52]. In many cases, malicious domains are registered in bulk at registrars with low reputation to reduce cost and avoid being taken down. In contrast, benign domains prefer reputable registrars, and many of them can be attributed to legitimate registrants using Whois information. Another type of malicious indicators relies on domain resolution patterns such as domain visitor diversity and reputation of resolved IPs [57] [43] [42] [39] [53] [21] [25] [22] [60] [52]. Finally, malicious indicators can be extracted from web content. These indicators are especially good at detecting phishing [77] [55], malicious JavaScript [46] [44] [48] [70], and malicious redirections [27] [48] [72]. Note that detectors that rely on this type of indicator can cover both abused domains (*e.g.,* `phishing[.]tld`) and malicious URLs (*e.g.,* `a-domain[.]tld/phishing-url`).

Classification techniques. Built upon the above indicators, detectors can leverage a couple of techniques to distinguish malicious domains from benign ones. The most popular technique is supervised machine learning, which is trained on a dataset of malicious and benign samples and classifies a domain as malicious or benign [21] [25] [22] [37] [53] [66] [53] [77] [51] [72]. Some other detectors first cluster similar domains based on specific indicators and then leverage supervised machine learning algorithms to classify a cluster as malicious or benign [23] [27]. Since it is sometimes challenging to curate a representative set of malicious samples, there are also detectors that use anomaly detection which can identify outliers deviating from known benign samples [60] [52]. The most widely used anomaly detection techniques include Support Vector Machine [29] and Isolation Forest [54]. Recently, graph-based analysis has attracted a lot of attention, and a couple of detectors apply graph mining on graphs built from domain resolutions and/or website redirections [57] [43] [49]. Finally, the detectors for domains hosting malicious JavaScript usually rely on signature-based approaches [46] [44] [70] [48].

Detectors deployed by *SV*. ***SV*** has implemented and deployed tens of malicious domain detectors in the past ten years. As indicated in the last column of Table I, these detectors have covered all detection scopes, types of indicators, and classification techniques described above since 2022, and they are continuously improved. These detectors are built on the same principles as the example detectors in Table I, and most of the features extracted from the malicious indicators share the same or similar definitions as the example detectors. Meanwhile, the classification techniques used by the ***SV*** detectors are mainly from popular open-source libraries, including scikit-learn, TensorFlow, and PyTorch. Finally, when evaluated using manually labeled datasets, the detectors of ***SV*** achieve comparable or better accuracy than the example systems. In summary, the diverse set of detectors deployed by ***SV*** significantly increases the representativeness of our study.

Third-party threat feeds used by *SV*. Besides the detectors built in-house, ***SV*** also ingests malicious domains from various external feeds. In this study, we focus on four most widely used third-party feeds, including VirusTotal, Spamhaus, URLhaus and Abuse.ch. Since not all engines on VirusTotal have equal accuracy and domain labels are known to be unstable [82], ***SV*** builds a proprietary algorithm to only ingest the highly likely malicious domains. In addition, malicious domains from all four feeds are checked against the allowlists built by ***SV*** (§II-D), and only those not in the allowlists are released as malicious. These publicly accessible threat feeds complement the detectors built by ***SV*** and greatly improve the representativeness of our study.

APPENDIX B
DESCRIPTION OF NINE TYPES OF USER COMMENTS

As described in §III-B5, the information in the user comments can be categorized into nine types. In this section, we describe each type of information.

No Comment. This is a special case where the *Comment* field in CRs is empty.

Claim Clean w/o Evidence. In these comments, users simply claim that the malicious domain is a FP without providing any evidence. Some typical comments include *"not a malware site"*, *"this domain is clean"*, *"category false, no malware"*, *etc*.

Ask for IOC w/o Evidence. In comments of this type, users ask ***SV*** to re-assess the domains and provide IOCs if the domains are kept malicious. For example, typical comments of this type include *"please evaluate this domain and provide your findings"* and *"would you please advise why this fqdn is associated with malware?"*.

User Own. Users claim that the reported domains are owned by them, *e.g., "this is our domain and it is not malicious."* and *"that site is for internal bussiness capacitation, our company have this site to demo our product to ours clients"*.

Valid Use. Users claim that they need to use the domains, *e.g.,* for business purposes. In many cases, users also describe why they need to use the domains, *e.g., "being blocked. [redacted] phones cannot communicate to [redacted] voip systems"* and *"user is unable to send/receive emails to the domain"*.

Clean OSINT. These comments mention that the reported domains are not classified as malicious by one or more open-source intelligence (OSINT). Users mention about 25 different OSINT sources, with VirusTotal, URLScan, AlienVault [13] and ANY.RUN being the most common ones. Example comments include *"I cannot find the threat related for this website from another sources like virustotal, shodan, forcepoint, symantec, ..."* and *"This URL has been ran through virus total and has come back clean with 0/90 hits."*. Since some OSINT services such as VirusTotal and URLVoid aggregates many online scan engines, we notice that some users claim that a domain is clean on VirusTotal and URLVoid even when there are one or two engines still tag the domain as malicious/suspicious/spam. For example, *"low hits in vt. please review"* and *"virustotal reports clean for the domain. urlvoid has 2 hits for bad domain: Avira and SCUMWARE. The scumware event is from 2018 and that URL is no longer there. The avira entry on virustotal returned clean so urlvoid could be using newer or older information."*. These cases are not excluded from this type of reason.

Malware Cleaned. In these comments, users mention that they are aware of website compromises that have been cleaned, *e.g., "the site had been compromised but has now been restored to original"* and *"malware removed from website"*.

Stale Detection. A few users report that a malicious domain was detected many years ago and the domain has expired, *e.g., "ownership has changed"* and *"your entry with this domain is from 2015"*.

Domain Content. Finally, the most common reason that users provide is that non-malicious content is hosted on the reported domains. We identify three sub-types of content:

- Meaningful content which indicates what content is hosted, *e.g., "web for educational people with problems like downs syndrome"*, *"automotive sites"*, and *"this is the web site of a local camera shop"*.
- Empty content which mentions that no content is hosted on the domain, *e.g., "not serving anything"*, *"site is no longer hosting any content"*, and *"content has been removed completely"*.
- Site redirection which claims that the domain redirects to a known website such as google.com and whatsapp.com, *e.g., "url is redirecting to google search engine hence raising the recategorization request"*, *"redirects to whatsapp"*, and *"it seems to be redirected to google site"*.

APPENDIX C
A RUNNING EXAMPLE OF IOBHUNTER

The running example in Figure 8 shows how trusted IOBs are identified for a real FP case using the transitive trust model. Given the detected FQDN `hillcrestfoods.com`, the search engine finds two sources (`datanyze.com` and `zoominfo.com`) that mention it (Search Round 1). Both sources attribute the FQDN to a legitimate company, and the content published on both sources is not from arbitrary public users. Therefore, the content is a valid IOB. We further check the trustworthiness of the two sources. Since neither is a root of trust, we iteratively search for them in search engines (Search Round 2). Based on the results, the root domains of the two sources are verified to be trusted by two government organizations, respectively. Since the Web content and subdomains of the two sources are owned by the same owner of their root domains, the trust of their root domains can be transited to the Web content and subsequently transited to the original detected FQDN. Now, trusted IOBs are found.

APPENDIX D
PROMPT OF IOBHUNTER

The CoT Prompt in `IOBHunter`

Consider the following search result and domain content. Analyze carefully to determine whether the domain is trustworthy or not.
Search target: {*entity*}
Full search context: *search result.*
Let's think through the following steps:
1. Does the content of search result exactly relate to {*entity*} or its root domain. If so, go to 2. Otherwise, return trusted_iob_found as no and stop.
2. Is the domain used by a legitimate app or owned by an legitimate organization? If so, got to 3. Otherwise, return trusted_iob_found as no and stop.
3. Is the web content on the source URL published by an arbitrary public user? If not, go to 4. Otherwise, return trusted_iob_found as no and stop.
4. Is the search target primarily a content-hosting or aggregation platform? (e.g.radar.cloudflare.com and cubdomain.com) If not, got to 5. Otherwise, return trusted_iob_found as no and stop.
5. Is the source domain an official government website (e.g. ending with .gov)? If so, trusted_iob_found is yes. Otherwise,

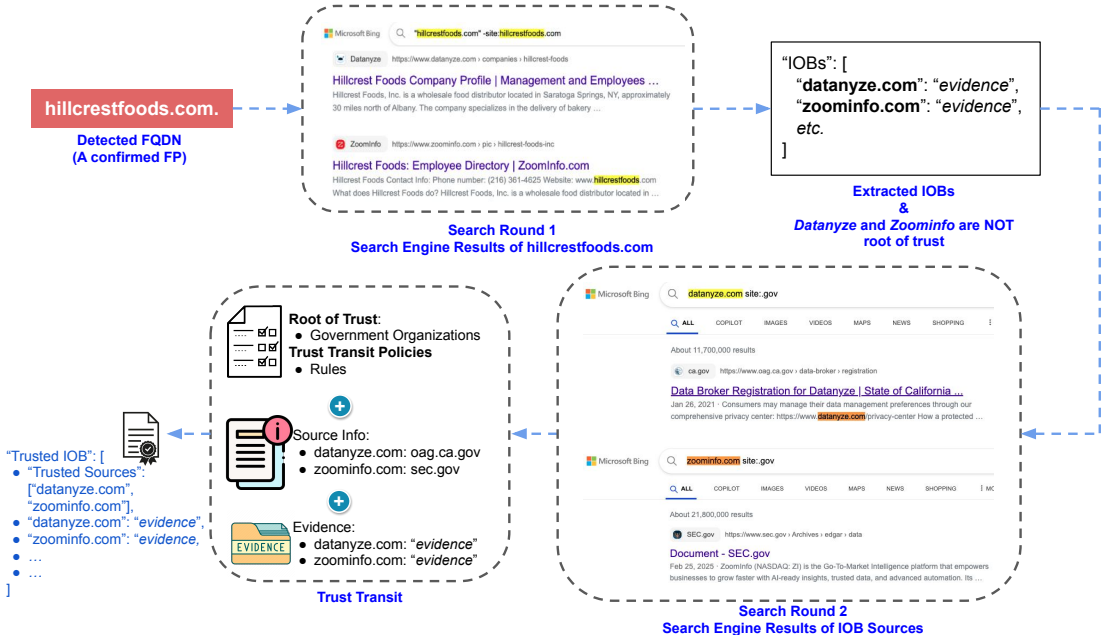

Fig. 8: A running example of IOBHunter.

trusted_iob_found is tbd.

Please return your answer in **strict JSON format** (use double quotes only, no trailing commas):
{{"trusted_iob_found": "yes", "no" or "tbd", "reason": "Concise explanation including one or more trust signals, such as: verified listing, official terms, no red flags, strong branding, etc." }}
Only return the JSON object.

To handle Step 3 and 4, we test a range types of known domains, including GitHub, social media platforms, WordPress-based websites and so on. For domains such as GitHub and major social media platforms, the process terminates at Step 4, as these are content-hosting platforms with contributions from arbitrary public users. In the case of WordPress, the outcome depends on how the site is hosted: for publicly hosted subdomains, the process halts at Step 4, while for self-hosted domains using WordPress, it halts at Step 3, as these are not considered general-purpose hosting platforms but may still involve user-generated content.

APPENDIX E
IOBHUNTER PSEUDO-CODE

Algorithm 1 shows the pseudo-code of IOBHunter.

Algorithm 1: IOBHunter

Input: Search Target: d, Full search context: s, Cache: C

Output: Trusted iob found: YES or NO

Initialize search round $i \leftarrow 0$;
Initialize search target list $T_0 \leftarrow \{d\}$;
while *termination conditions not met* **do**
  foreach *FQDN* $f \in T_i$ **do**
    if f *is root or shares ownership with root* **then**
      Add f and/or root(f) to T_{i+1} ;
  Retrieve search results R_i for all $f \in T_i$;
  foreach *result* $r \in R_i$ **do**
    if $d \in C$ **then**
      return cached value for d ;
    if r *mentions* f ***and*** f *is used by a legit app/org* ***and*** *content is not from public users* ***and*** f *is not a host/agg platform* **then**
      if f *is an official government site* **then**
        Cache C $(d, \texttt{YES})$;
        return YES ;
      Add IOB source FQDN to T_{i+1} ;