Practical Real-time Detection of IPv4 Record Classical Domain Hijacking at Scale

Janos Szurdi
Palo Alto Networks
Budapest, Hungary
szurdi.janos@gmail.com

Mohammad Ghasemisharif
Palo Alto Networks
Santa Clara, USA
mghasemishar@paloaltonetworks.com

Reethika Ramesh
Palo Alto Networks
Santa Clara, USA
reramesh@paloaltonetworks.com

Zhanhao Chen
Palo Alto Networks
Santa Clara, USA
zhachen@paloaltonetworks.com

Ruian Duan
Palo Alto Networks
Santa Clara, USA
rduan@paloaltonetworks.com

William Melicher
Palo Alto Networks
Santa Clara, USA
bmelicher@paloaltonetworks.com

Daiping Liu
Palo Alto Networks
Santa Clara, USA
dpliu@paloaltonetworks.com

Abstract

One of the most dangerous cyberattacks is classical domain hijacking, where an attacker takes control of the administration of a domain's existing DNS records, modifying them, and sending visitors to malicious destinations. Domain hijacking can have a disastrous impact on both domain owners and their users, as criminals can direct all users to Man-in-the-Middle, phishing, and a wide array of other attacks. In recent years, researchers have analyzed and built detectors for various types of DNS hijacking attacks, including domain shadowing, exploitation of mistyped domains, and abuse of dangling records. However, only two studies on classical domain hijacking exist, failing to provide the community with a method to identify domain hijacking precisely and in time. This paper explores the feasibility of real-time classical domain hijacking detection for IPv4 records and characterizes its prevalence in the wild.

We designed a framework that can process hundreds of millions of new records to identify a few dozen hijacking records or to analyze live DNS traffic from customers of a large cybersecurity company in around ten minutes. The framework leverages hundreds of terabytes of passive DNS logs and geolocation databases. It also utilizes active measurements such as WHOIS and web crawls to collect additional information about potential hijacking records. We evaluated the framework on our labeled datasets and additionally tested it in a large DNS security provider's network for over a year, finding multiple new domain hijacking attacks. Our work provides the first method for detecting domain hijacking with high precision, in real-time, and tested in a production environment.

ASIA CCS '26, Bangalore, India

ACM ISBN 979-8-4007-2356-8/26/06
https://doi.org/10.1145/3779208.3807492

CCS Concepts

• **Security and privacy → Network security**.

Keywords

Hijacking, Domain Names, DNS, Artificial Intelligence

ACM Reference Format:
Janos Szurdi, Mohammad Ghasemisharif, Reethika Ramesh, Zhanhao Chen, Ruian Duan, William Melicher, and Daiping Liu. 2026. Practical Real-time Detection of IPv4 Record Classical Domain Hijacking at Scale. In *ACM Asia Conference on Computer and Communications Security (ASIA CCS '26), June 1–5, 2026, Bangalore, India.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3779208.3807492

1 Introduction

DNS hijacking encompasses a wide range of attacks, including domain hijacking [2, 33, 45], abuse of mistyped [72] or dangling records [25, 44, 80], malicious resolvers [24, 39], DNS spoofing, and Man-in-the-Middle (MitM) attacks [55]. Among these, domain hijacking poses one of the most severe risks, allowing attackers to redirect all traffic and visitors to malicious servers. One notable example is when criminals hijacked multiple domains of a large Brazilian bank [12], exposing millions of users to phishing and malware. The attackers gained administrative control of the bank's domain names by compromising the bank's DNS service provider using an unknown initial attack vector that could range from exploiting a vulnerability to a successful phishing attack.

In this work, we consider domain hijacking to be when attackers gain control of the administration of a domain's DNS records, allowing them to delete, insert, or modify records. Attackers hijack domains by stealing the registrant's credentials at a domain registrar or a DNS service provider or compromising either the registrar, registry, or DNS service provider. In contrast, DNS hijacking encompasses any method, including domain hijacking, that results in a DNS client accepting a DNS record crafted by an attacker.

While most of these attacks have been adequately addressed and researched in-depth, domain hijacking has not been studied

systematically until recently [2, 33], and no effective solution has been provided. Akiwate et al. [2] provide insights about domain hijacking by studying it retroactively, where they employ features that are only available post factum and rely on manual investigation, making their approach unsuitable for timely detection. Houser et al. [33] presented the first work to detect domain hijacking automatically but did not achieve performance suitable for real-world use, which requires tight false positive (FP) rates in addition to high recall. Improving on their work, our systems aim to achieve four novel goals: 1. High-precision classification suitable for autonomous deployment at scale. 2. Real-time detection to protect users in time. 3. Longitudinal deployment to a production system for a real-world test of detectors. 4. Continuous monitoring and analysis of hijacking attacks in the wild.

The research community struggled to propose a solution for domain hijacking, as designing a detection pipeline comes with multiple challenges. First, publicly shared domain hijacking attacks are rare, providing limited data available for training classifiers. Second, even though attacks are rare, a detection pipeline must process hundreds of millions of records daily, making high-precision detection difficult. Third, from the perspective of DNS logs, domain hijacking and changing hosting provider can look very similar.

We developed the **A**utomated **C**lassical Hijacking **ID**entification Framework (ACID) to fill the gap for classical domain hijacking detection. We demonstrate ACID's capability by focusing on A record hijacking, one of the most commonly used and hijacked DNS record types [33]. Using ACID, we built two detection pipelines to answer four questions related to the aforementioned goals. 1. Is it possible to detect domain hijacking at scale with a performance suitable for autonomous detection? Given that previous work [33] has not been able to achieve sufficient performance metrics even on labeled data. We aim at a more ambitious goal to solve the problem at scale such that it can be deployed in a production environment. 2. Is it possible to detect domain hijacking in real-time? A detector is only useful if it can protect users in time. Therefore, we ask what are the trade-offs in creating a real-time detector with a small enough delay to protect victims. 3. What is the extent of domain hijacking? Is domain hijacking really as rare an occurrence as previous work [2, 33] hinted? 4. How are hijacked domains utilized?

After researching public sources, we found only 48 classical hijacking attacks targeting 44 domains, which also had historical DNS data to understand them. Although our findings are comparable in quantity to what other researchers have found [2, 33], it is not enough to train and test a machine learning classifier. To solve this, we simulated domain hijacking attacks based on our set of known domain hijacking cases, providing a hybrid labeled dataset of millions of DNS records—a mix of newly observed DNS records and simulated hijacking attacks. The hybrid dataset is systematically designed to cover a wide variety of possible domain hijacking attacks beyond what the few ground-truth examples could offer, allowing ACID to detect realistic novel attacks.

We then train a machine learning classifier on our hybrid labeled data using 75 mostly novel features. Our classifier achieved over 99.7% area under the precision-recall curve. Depending on the precision-recall tradeoff, our model detected between 54-92% of known hijacking attacks, a commendable performance given that these attacks often happened a long time ago, where our DNS dataset might not be complete.

Although the classifier performed much better on labeled data compared to previous work [33], we also assessed its performance in a production environment where data distribution constantly changes and hundreds of millions of records need to be processed daily. In such an environment, 99% precision can lead to tens of thousands of FP detections every day. Therefore, we carefully designed an enrichment and analysis module that runs active measurements (e.g., Web and WHOIS crawling), inspects JavaScript execution, compares certificates, and leverages a language model to understand screenshots. ACID's in-depth analysis of detections allows it to preemptively remove FPs if evidence is found and block hijacking records otherwise, even if the attackers target applications other than the Web. Additionally, the analysis module enables ACID to distinguish inconspicuous (where the website looks the same after hijacking) and conspicuous domain hijacking.

We have deployed our offline detection pipeline since January 2024, blocking DNS responses containing hijacking records for Palo Alto Networks (PANW) customers. In this paper, we analyze domain hijacking detected between January 1, 2024 and April 30, 2025. During this period, our pipeline processed 90.7 billion newly seen A records, an average of 186.5 million daily. After classification and further analysis, ACID blocks the remaining 5,121 records for PANW customers, a mere 11 domain hijacking records per day. The number of detections is seven orders of magnitude less than the number of records analyzed. Although domain hijacking is relatively rare, we find that it occurs regularly (Figure 7) and more frequently than expected, where even one attack can have a calamitous impact.

Analyzing our detections, we identified several novel, previously unreported cases of domain hijacking. We find hijacking targeting domains of a political party, a large Internet service provider, a popular utility company, a news website, and a research center, among others. Attackers redirected users to various targets, including phishing, gambling, and defaced websites. Our findings on how domain hijacking is utilized largely align with previous work [33], except for several instances where hijacked domains are redirected to Asian gambling sites, likely illicit in their country of origin.

In summary, we present ACID, the first design to detect domain hijacking without causing issues to users in a real-world deployment. To help the research community at large, we make several contributions. First, we share domain hijacking attacks found by ACID and plan to regularly report new findings, as one of our biggest challenges in building this system was obtaining labeled data. Second, we share the known hijacking records and all related public sources we collected. Third, we share the simulated hijacking attacks we have created. Finally, while the source code and data are proprietary, we provide a detailed blueprint in this paper of how ACID works, enabling researchers to reproduce our work. The shared data can be found at https://github.com/jszurdi/domain-hijacking. We hope that by demonstrating that domain hijacking detection is possible, sharing our research and reporting on detected domain hijacking cases will inspire future work to improve upon our method and stimulate interest in further studying these attacks.

2 Background & Related Work

2.1 DNS records

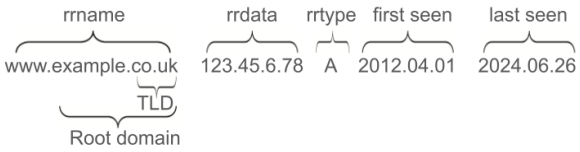


Figure 1: Example simplified DNS record

Our analysis in this paper is centered around DNS records that are also called resource records and we abbreviate them as RRs. We use a simplified version of RRs–as shown in Figure 1–that contain the rrname, rrdata, rrtype, first seen date, and last seen date. The first seen and last seen dates are calculated according to PANW's passive DNS dataset as described in §3.1.

The rrname in an RR has a top-level domain (TLD). For example, in Figure 1 the TLD is *.uk* for the United Kingdom. However, *.co.uk* is a public suffix where registrants can register domain names. For simplicity, we use the abbreviation TLD for such public suffixes [59]. Domain names registered directly under public suffixes (*example.co.uk* in the example) are often called registered, apex, or root domains. We use the term root domain or often just *root*.

2.2 DNS Hijacking Taxonomy

The Domain Name System is a foundational and core component of the Internet translating human-readable domain names to IP addresses that in return machines can use to locate other machines. Given its vital role in guiding users to online services, attacks targeting DNS can be catastrophic

Researchers have studied a wide variety of attacks targeting DNS. As previous work has discussed them in detail [2, 33], we just provide a brief overview. In Figure 2, we created a categorization of the most important attack types that can result in DNS hijacking.

DNS Hijacking One of the most studied attacks is called DNS cache poisoning, where attackers trick DNS servers to store and use malicious DNS records[3, 5, 15, 20, 31, 36, 54, 58, 62, 74, 78, 79, 83]. The latest research has shown that DNS cache poisoning is also possible through side-channel attacks [48]. Another widely researched attack is MitM, where DNS responses are modified during their journey to resolvers [55, 61, 73]. An additional popular area of research is the analysis of how MitM DNS attacks are employed for censorship [1, 8, 9, 32, 42, 46, 49, 51]. Furthermore, some malware can compromise the DNS settings of victim machines to point to malicious DNS servers that craft malicious DNS responses [19, 24, 39, 69]. A large body of work has found problems related to dangling DNS records—yet another method for attackers to compromise DNS records—where domain owners have neglected the maintenance of the DNS records of their domains, allowing attackers to take control of domains or IP addresses where these records point [4, 25, 44, 80–82]. Similarly, a variant of typosquatting [65] where mistyped DNS records, for example name server records, point to a domain name not under the control of the domain owner [72]. Bitsquatting [52] can also lead to DNS hijacking, as bit-flip errors have a small probability of redirecting users to an attacker controlled domain.

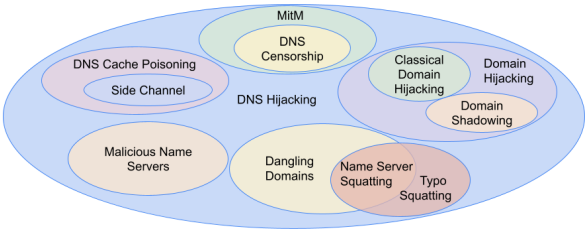


Figure 2: DNS hijacking classification.

Domain Hijacking *Domain hijacking* is when an adversary takes control of the administration of the DNS records of a victim domain name. Domain hijacking allows attackers to add, remove and modify DNS records of the affected domain names. While dozens of papers dissect various DNS attacks and their defenses, minimal research [2, 33, 45] has studied *domain hijacking*.

Cybercriminals can hijack domains by gaining access to entities that participate in the administration of the domain name. Criminals can compromise registrants' accounts at registrars or at DNS service providers. They can compromise these accounts through phishing or data breaches. Additionally, domain hijacking is possible if attackers breach registrars or DNS service providers (or even registries) directly, which potentially affects a large number of domains. By modifying existing records, attackers can divert users to malicious servers and expose them to a variety of malice including defacement, phishing, scam, MitM, and drive-by-download.

Domain Shadowing vs Classical Domain Hijacking While classical domain hijackers modify existing records to target users of compromised domains, in the case of domain shadowing, attackers only create new subdomains to avoid disrupting the compromised domain's normal operations and remain hidden longer. Liu et al. [45] developed a machine learning approach based on features of subdomain usage, hosting, activity, and lexical properties. From their work, only the hosting features are relevant for detecting classical domain hijacking, as shown in Table 4.

Houser et al. [33] is the only previous work that attempted to develop a detector for classical domain hijacking, but acknowledged limitations in practical performance and the need for further improvements. Unlike our work identifying hijacking RRs, they focused on detecting if a domain was hijacked altogether; thus, they developed features for different record types, including IPv4, mail server, and name server records. As our features aim at a deeper understanding of new A records' connection to a domain's history, only a fraction of our features overlap with theirs (Table 4).

Akiwate et al. [2] conducted a study identifying domain hijacking retroactively. Their approach is not employable for timely detection, as they focus on identifying transient deployments that do not persist beyond three months, and they also require manual analysis of suspicious deployments,

ACID addresses the research gap left by prior work [2, 33], which provides no adequate solution for the timely detection of classical domain hijacking, by introducing a real-time approach to block classical domain hijacking records, validated in PANW's production environment.

Table 1: Summary of ACID's adversary setting coverage.

Target Application	Website and Certificates Changed	Unchanged	None
Web	Yes	(No)	Yes
Other (FTP, email, ...)	Yes	No	Yes

Table 2: Record naming at different stages of our detection pipelines.

Stages of Records	Description
New records	Domain rrdata (e.g., IP) pairs we have never seen before.
Candidate domain hijacking	RRs not in the historical /24 subnets of the root.
Predicted domain hijacking	RRs our model predicts as domain hijacking.
Potential domain hijacking	RRs we start blocking for users and monitor for FPs.
Domain hijacking	Records that we keep as domain hijacking.

Compared to previous work on classical domain hijacking [33, 45], which focuses only on evaluating machine learning classifiers on labeled data, ACID is a complex and practical framework that allows automatic domain hijacking detection at scale, uses novel features, enriches detections through active measurements, and performs in-depth analysis for both conspicuous and inconspicuous domain hijacking. Furthermore, ACID is "vetted in action", as our pipelines have been deployed for over a year for one of the largest security companies. In addition, the production deployment enables us to analyze domain hijacking in the wild in §4.2.

Malicious Domain Registrations We do not consider work on the detection of malicious domains using machine learning [29, 30, 53, 57, 63, 66, 77] as they aim to solve a fundamentally different problem. While they might use similar data sources, our feature construction differs greatly and they do not conduct comparisons between new and historical DNS records of the same domains.

2.3 Classical Hijacking Adversary Model

We assume an adversary who controls the administration of a domain's records and can modify DNS records at will, as described for classical hijacking. After changing the DNS record, the adversary extracts value by sending users to attacker-controlled servers. To be successful, they need to set up applications on these servers as expected by users, for example, Web, email, or FTP servers. We assume an attacker who does not have control of the application server and the private key of the certificate on the server.

ACID focuses on detecting A record hijacking and provides a blueprint for future work on detecting classical domain hijacking for other record types. While including IPv6 (AAAA) records seems closest to detecting A records, we leave this for future work due to the challenge of IPv6 distance calculation; for instance, some end users might get access to entire /64 IPv6 subnets consisting of 2^{64} addresses, while mobile providers issue individual IPv6 addresses. To filter potential FPs and understand attacker operations, ACID actively collects certificates and websites for records predicted to be hijacking by ACID's machine learning model. Table 1 shows that if an attacker changes the certificates or the website on the hijacking server, or if they do not exist, then ACID can decide not to discard the predicted hijacking record as an FP and block it for customers (details in §3.4). If a careful attacker ensures that the original website and certificates from before the attack are returned, then ACID will discard the predicted hijacking records as FPs. To detect such adversaries targeting non-web applications, future work needs to collect data specifically for those applications. If the adversary targets the Web and does not change the website or certificates, then they can only steal data if HTTPS was not used before, which is increasingly rare nowadays [28]. Without access to the server, an adversary cannot provide a valid signature for modified content without changing the certificate.

Many of our features describe how a target domain behaved in the past and how the hijacking IP relates to that history. Changing the history of target domains would be expensive for attackers. They could also select IPs for hijacking that look similar to historical IPs of the target domain and avoid modifying name server records. However, these would increase the cost or difficulty of the attack, for instance, if the target domain uses a local small hosting provider or has its own IP range. Depending on the attacker's entry point, it might not be feasible for the attacker to not change the NS records for a successful domain hijacking attack. In Appendix A.2, we discuss the capability of ACID to detect domain hijacking, even if an attacker manages to use an IP address with an ASN/ISP present in the history of the hijacked domain.

3 Methodology

We follow six steps to detect and understand domain hijacking: First, we collect and study known domain hijacking samples from blogs and reports. Second, we simulate hijacking to create a sufficiently large labeled dataset to train an ML model. Third, we train and evaluate the ML model using both simulated and known hijacking data. Fourth, to materialize ACID, we design an offline detection pipeline that leverages the ML model cost-effectively in a production environment, processing all new records in batches (e.g., once a day). Fifth, we develop a real-time pipeline capable of identifying hijacking within ten minutes. Both pipelines are integrated into PANW's security solution, blocking DNS responses containing hijacking records and correcting plausible false positives retroactively. The real-time pipeline is functionally the same as the offline one, but requires complex engineering, which we discuss in the Appendix A.4. Finally, we analyze detections after running our detectors for over a year.

New records are RRs that we see for the first time in passive DNS. The offline detection pipeline processes all new records daily (§3.4). Differently, when we calculate root domain features in Table 4, we consider the RRs seen in the last three days for a domain as new. Table 2 summarizes the DNS record terminologies across the stages of our offline and real-time detection pipeline. Additionally, hijacking records that are manually verified by us or someone else are termed as *verified* or *true* domain hijacking records.

3.1 Data Sources

To make domain hijacking detection feasible at scale, we leverage a combination of passive and active datasets.

Passive DNS dataset The core dataset is a passive DNS dataset maintained at PANW. Passive DNS refers to logs of DNS requests

and responses collected from multiple vantage points. The passive DNS dataset contains anonymized data from Farsight's DNSDB [23] and from customers of PANW. We utilize a derivative of this dataset called unique RR, which stores <rrname,rrtype, rrdata> triplets with first and last seen timestamps in BigQuery [10]. As of May 2025, this continuously growing dataset contained 692 billion unique RRs, totaling 212 Terabytes.

pDNS API We utilize PANW's pDNS API service to efficiently query the pDNS history of a domain name. The pDNS API is based on a wide-column, key-value NoSQL database (BigTable [11]). We leverage pDNS API for fast key-based lookups (e.g., based on the rrname) as described in §A.4. However, it is not efficient for complex large-scale aggregation jobs required for feature calculation.

New records We identify newly observed DNS records by checking if a <rrname,rrtype,rrdata> triplet does not exist in our unique RR data. We use new records for training (§3.3) and in the offline detection pipeline (§3.4).

IP address geolocation dataset Apart from the pDNS dataset, IP address geolocation is crucial for our classifier. We use IP2location's dataset [35] which provides us with estimated geographic locations of IP addresses, including country, Internet service provider (ISP), and autonomous system number (ASN). We additionally enrich it with a custom subregion field (e.g., Europe for England).

WHOIS dataset We use WhoisXML's WHOIS database [75] to get domain registration information, primarily the domain registration date. We also scrape and parse WHOIS data using the python-whois [60], to complement WhoisXML in cases where a record is stale or missing.

Web content and certificate dataset We scrape each suspected hijacked domain using a custom instrumented Chrome browser, that allows IP address modification when visiting the domain. This change only redirects domain resolution to a predetermined IP address without modifying the web traffic. This is a necessary feature because we test multiple IP addresses for each domain, some of which require hostname-to-IP address mapping. During each crawl, we collect screenshots, HAR files, and certificates.

Malicious domain dataset We utilize PANW's malicious domain database to better understand the domain's current and historical IPs related to hijacking. This database integrates their in-house detection systems with third-party threat intelligence sources such as VirusTotal [71].

Known hijacking dataset For ground truth validation, we compiled known domain hijacking cases from 2014 to 2021 using blogs, reports, and related research. We searched for these cases from our passive DNS dataset, ensuring that each domain has a history for the rrname before hijacking. We identified 152 A records, 88 root IP address pairs, 44 root domains, and 62 attack dates. This size aligns with datasets collected by previous work [2, 33], such as the 79 "instances" of hijacking reported by Houser et al. [33]. We grouped these records into hijacking campaigns based on three criteria: IP address similarity (same /24 subnet), geographic colocation (matching country, ISP, and ASN), and temporal proximity (hijacking dates within seven days). This categorization yielded 48 distinct hijacking campaigns.

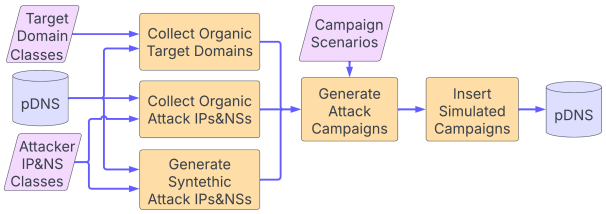


Figure 3: Overview of domain hijacking simulation design.

3.2 Simulation

Similarly to previous research [2, 33], we found validated examples that we can use as labeled data is limited, amounting to a mere 152 records of 44 root domains. Thus, we systematically simulated domain hijacking to create a larger and more diverse labeled dataset. We designed our simulation methodology based on analyzing what validated domain hijacking attacks look like in pDNS, previous work [2, 33], and consulting with PANW experts. Our goal was to capture both previously known hijacking attacks and also vary parameters such that we can detect future hijacking attacks that appear different in pDNS but are realistic.

Figure 3 describes how we create different simulated hijacking attacks with complexity that varies in four dimensions: target domain class (the hijacked domain), attacker IP address, attacker name server (NS) classes (important detection feature), and the hijacking campaign scenario. Our approach consists of three steps. First, we select target domains, attacker IPs and name servers to construct our domain hijacking records. Second, we create different domain hijacking campaigns by assembling simulated hijacking records. Finally, we insert the simulated records into our pDNS dataset to train a machine learning classifier.

In Figure 3, "organic" describes a target domain, IP, or name server used for simulation that has an established history in our pDNS dataset. For attacker IPs and NSs, we additionally randomly generate synthetic rrdata and remove those that exist in pDNS.

We categorize how challenging domain hijacking detection would be for target domains based on the richness of their pDNS history. Many known hijacking attacks target domains historically resolving only to a single IP address within one country, where detecting a suspicious change is relatively easy. However, it is harder to detect hijacking in domains that use multiple providers and resolve to thousands of IPs in dozens of countries. Therefore, we group target domains into three classes: easy, medium, and hard. We classify target domains by the number of IP country codes and the name server root domains. We also ensure to include domains from both gTLDs and ccTLDs, with different proportions of IP countries matching the ccTLD and self-hosted name servers. Specific parameters are discussed in the Appendix A.2. We randomly select target domains from the combination of domains owned by the PANW's customers, popular domains and sensitive domains using the above factors. Popular domains are the union of Tranco top domains [41] and domains frequently resolved by PANW's customers. The sensitive categories include financial services, government, military, web-based email, and cryptocurrency.

To train a robust classifier, we consider a diverse set of nameservers and IP addresses for our simulated hijacking records. We

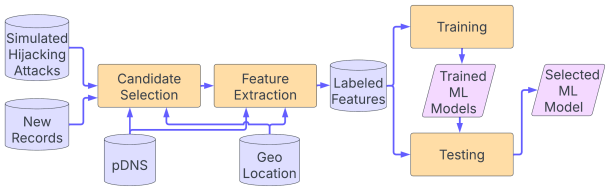


Figure 4: Overview of training a domain hijacking model using simulated data.

consider the average record age, ratio of malicious root domains hosted, and number of root domains and TLDs. For nameservers, we additionally assess if they are malicious, the proportion of self-hosted instances (NS root matches rrname root), and proportion of times where the NS TLD matches the rrnames. We create a large number and wide variety of records combining organic rrnames and organic or synthetic rrdata from different classes to form new hijacked records. We additionally categorize these records, by comparing the rrdata to the rrname history, to see if an IP's geolocation data has already been seen in the rrname's history or whether the country of the IP matches the ccTLD of the rrname. We created six different types of campaigns from the simulated attacks ranging from small-scale attacks—common for previously known hijacking cases—targeting single domains leveraging one IP address to large campaigns targeting dozens of domains across multiple IP addresses. Each campaign type, detailed in the Appendix A.2, varies across key dimensions such as the number of target domains, attacker IPs and nameservers, and their distribution per target domain.

Finally, we insert these simulated records into our pDNS dataset, enabling our pipeline–up to feature extraction–to process them identically to newly observed DNS records.

3.3 Training a classifier using simulated hijacking records

Figure 4 explains how we inject the simulated hijacking records into passive DNS as positive hijacking examples to train an ML classifier. We consider all new records observed in passive DNS between 2023-09-02 and 2023-09-07 as the set of negative non-hijacking example. Although there may be a small number of hijacking records among these negatives, given the hundreds of millions of daily new DNS records, their expected proportion is negligible (<0.001%), making the impact of such label noise on model training minimal.

We process labeled records similar to our offline detection pipeline described in §3.4. Given the high daily volume of new records, we employ a candidate selection filter, shown in Figure 4, to optimize processing costs. Our intuition behind candidate selection leverages the observation that hijacking IPs are typically unrelated to the domain's historical IPs, a pattern consistent across all known hijacking records in our dataset. The candidate selection removes all records where the /24 subnet matches any /24 subnets in the history of the rrname's root domain and subdomains. Additionally, we remove records where the rrnames are younger than six weeks, called Newly Observed Hostnames (NOHs). We filter NOHs because they fall under the purview of domain shadowing [45] as opposed to classical domain hijacking. This filter alone removes the vast majority, ≈98%, of all new records.

Table 3: Table listing the number of training and testing example after prefiltering and candidate selection stages.

	Training set	Testing set
Simulated hijacking RR #	180,990	45,216
New RR #	6,859,811	1,714,985

Table 4: Table of features. * denotes that we calculate four statistics for the features min, max, mean, and standard deviation. ** marks that we consider it as a new record if it was first seen during the past one week. Y (all) means that all listed features are novel. (Y) means that the features was not used to detect classical domain hijacking but it was used differently for other detectors in previous work.

Feature Category	Feature	Novelty
Historical IP addresses (20 features using A RRs)	Number of root domains*, Number of TLDs*, Average RR age*, Number of malicious root domains*, Proportion of malicious root domains	Y (all)
New IP address (7 features using A RRs)	Number of root domains, Number of new root domains**, Number of TLDs, Average RR age, Number of malicious root domains, Proportion of malicious root domains, CC matches domain TLD	Y (all)
Comparison of new to historical IP addresses (28 features using A RRs)	Difference in Number of TLDs*, Difference in Number of malicious root domains*, Difference in proportion of malicious root domains*, Is IP ASN new?, Is IP country new?, Is IP ISP new?, Is IP subregion new	Y (all)
	Difference in Number of root domains*, Difference in average RR age*, Difference in IP address integer value*	(Y) [45]
Root domain features (20 features using all RRs)	Number of RR types in new RRs**, Number of root domains seen in the domain's new NS records**, Number of new subregions associated with the A records**, Age, Number of subdomains, Number of /24 subnets, Number of countries of IP addresses, Number of subregions of IP addresses, Number of ASNs of IP addresses, Number of countries of IP addresses matching ccTLD, Number of nameservers' root domain, Number of self-hosted nameservers, Is TLD a ccTLD?	Y (all)
	Number of new RRs**, Number of IPs seen in the domain's new A records**, Number of new ISPs seen associated with the A records**, number of new countries associated with the A records**, Number of IP addresses, Number of ISPs of IP addresses, Number of nameservers	N [33]

Table 3 demonstrates that our process of obtaining labeled data resulted in thousands of additional samples even after candidate selection compared to previous work [33]. The scale and diversity of our labeled dataset enhances our detector's reliability. For example, with 1.7 million benign samples in our test set, we can better calibrate the trained classifier for high precision to minimize FPs in production. Furthermore, we have increased confidence in its ability to detect domain hijacking, especially running a final model validation step against known hijacking records that were not used in the prior training and evaluation process.

After candidate selection, we calculate 75, mostly novel, features from the combination of passive DNS and geolocation datasets. We calculate four feature categories describing historical IPs, the new

Table 5: Performance metrics for various classifiers

Classifier	Precision	Recall	F-1	AUC-ROC	AUC-PR
Random Forest	**0.9899**	**0.9686**	**0.9792**	**0.9998**	**0.9974**
Gradient Boosting	0.9795	0.9504	0.9648	0.9997	0.9935
AdaBoost	0.9703	0.9604	0.9653	0.9998	0.9939
Decision Tree	0.9597	0.9594	0.9595	0.9792	0.9218
MLP	0.9408	0.9242	0.9324	0.9993	0.9805
KNN	0.9088	0.8493	0.8780	0.9752	0.903
Logistic Regression	0.8519	0.7958	0.8229	0.9960	0.9025

candidate hijacking IP, the comparison of the candidate IP with historical IPs, and the target domain itself. For the applicable features in Table 4, we calculate four statistics: minimum, maximum, mean, and standard deviation. We define RRs, IP addresses, or domains as new if they have been first observed within the past seven days. Differently, we consider IP geolocation (country code, ISP, ASN, and subregion) new if it differs from the root's historical geolocation. ACID calculates features about all types of RRs, not just A records, to understand domain hijacking, for example, the number of RR types or the number of new name server root domains observed.

After feature extraction, our preprocessing pipeline prepares the samples for training by taking feature vectors and removing any non-numerical values or duplicates to ensure that the training samples do not leak into the test samples. As some ML models cannot handle missing feature values, we adopt a strategy called missing value imputation, where we replace the missing values with the mean of the data observed for that feature. The pipeline also standardizes the features by rescaling them so that they have a mean of zero and a standard deviation of one. This is important as some ML algorithms are sensitive to the scale of the input data.

We use the prepared feature vectors and their corresponding labels to train and compare a set of ML models (see Table 5). We perform a five-fold cross-validation to test the performance of each model. At each fold, 80% of the data is used for training and 20% of the data set aside for testing. At the end of the cross-validation, the average performance of the five folds is reported as the model's performance. For parameters, we set 200 estimators for Random Forest, Gradient Boosting, and AdaBoost, and set five neighbors for KNN. We leave rest of the parameters as default values in scikit-learn. We also tested a multilayer perceptron classifier with two hidden layers with sizes of 64 and 32 and a ReLU activation function. We found that Random Forest had the best performance compared to other models. The results are presented in Table 5.

Once we obtained the best performing model, we performed threshold tuning to find the threshold that has the best precision performance. We randomly split data into 90% for training and 10% for testing. We train the Random Forest model on 90% of the data and search for the threshold that results in the highest precision and use the threshold on the test data (10%) to calculate expected precision and recall values. This threshold is used in the pipeline to detect domain hijacking.

3.4 Offline Pipeline Design

We designed the offline detection pipeline (Figure 5) to process newly observed DNS records in batches from all sources available to PANW. In contrast, the real-time pipeline is designed to process DNS records immediately upon observation, but only from PANW's customer traffic.

Figure 5 provides an overview of how the offline domain hijacking detection pipeline works. The input of the pipeline is the set of newly observed DNS records. First, during prefiltering the pipeline removes records that are unnecessary: records with invalid fields (e.g., domains containing invalid characters, IPs that have invalid values), records that have values that only work on local internal networks (e.g. internal domains and private/reserved IP ranges), and record types that we do not use in our pipeline. Next, due to the daily volume of tens to hundreds of millions of new records, we employ the candidate selection module described in §3.3, to identify likely hijacking cases reducing both FPs and computational costs. For selected candidate records, we extract features as detailed in §3.3 and process them through our pretrained model to determine the likelihood of whether a domain is hijacking. Despite the model's high confidence levels, production-scale processing and data distribution changes can still lead to FPs.

Enrichment and Analysis of Detections For predicted hijacking records, we gather additional information to reduce FPs and understand them better. The number of predicted hijacking records are orders of magnitude lower than new DNS records making further data collection economically and computationally feasible. We collect WHOIS data and active crawls of the website to collect screenshots, the landing page DOMs, loaded resources, certificate chains, and HAR files. We use a custom, automated headless Chromium browser that performs DNS mapping, which maps the given domain with the provided IP address without performing a DNS resolution. This helps us to visit the website at a specified destination IP. We also modify the User-Agent string to make it less probable that we are identified as a bot. We utilize the additional information to make a final decision whether we believe the record is truly hijacking or not and send the record to a datastore to be blocked for customers. Each piece of data plays a different role. WHOIS data helps identify newly registered domains, indicating ownership changes rather than hijacking incidents.

We visit the target domain website three times using three IP addresses and for each, we map the target domain to the IP address before crawling: 1) the new IP address that was detected as hijacking and 2) the two most recent historical IPs before the hijacking. After each visit, we collect the SHA256 hashes of the final DOM content, the resource URLs, content loaded during web crawl, screenshots of the landing page, and the certificate chains. Then, we conduct a comparative analysis between the content downloaded from the hijacking and the historical IP addresses. If there is a similarity between them, we classify the record as not hijacking. If the certificate chain received from either of the historical IP addresses is the same as the certificate hosted on the hijacking IP address, then the website owner likely initiated the change in the record. Similarly, we identify benign website migrations by comparing loaded resources and DOM structures across the hijacking and historic IP addresses. For comparison of screenshots, we use average hash (aHash) [13] with a Hamming distance of less than 10 to capture similar or identical screenshots. Based on our experiments, this can ignore the differences caused by small dynamic content such as advertisement placements.

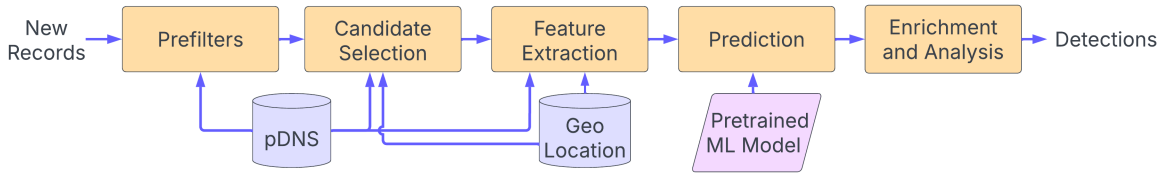


Figure 5: Design of the offline domain hijacking detection pipeline.

Table 6: Screenshot categorization criteria for Gemini.

No content	Empty page, Error page, Short text only, Parked, Under development, Default page, Other Website without content, Loading page, Verify you are human/CAPTCHA page
Normal content	Business website, Personal website, Other Website with content
Potentially interesting content	Webshop, Login page, Download page, Streaming, Payment page, Form page
Questionable content	Gambling, Adult, Torrent, Defaced webpage, Scam page, Phishing page

Generative Model to Understand Screenshots We found that aHash comparison is often insufficient, for example, when domain owners change hosting provider and update their website at the same time. Such a change can create new DNS records that look like domain hijacking and also change a website's appearance (Example shown in Figure 12 in §A.2). To address this issue, we use Google's Gemini multimodal AI [26], a generative model, to process and compare the screenshots captured by ACID. We use Gemini to identify such cases using two prompts (full prompts in §A.5):

> *Tell me if the topic of the two images of the website are similar, answer with "yes" or "no".*

> *List at most the top four categories that apply to the image of the website from these categories: .....*

Table 6 lists all the categories we ask Gemini to identify for our screenshots. We set the parameters [27] top-K to one and temperature to zero, forcing Gemini to select the highest probability token, leading to less variability in its response.

We sample 200 cases, comprising 600 screenshots, for manual analysis and find that Gemini performs surprisingly well. Gemini had 99.7% accuracy in categorizing screenshots and 95.8% accuracy comparing screenshots. The only significant issue Gemini faced in deciding if two screenshots had a similar topic is when one of the screenshots is error, empty, or short text and the other is normal content. Thus, we do not include such cases for comparison. Notably, we exclude detections where hijacked pages showed content, but the original pages are empty, as these are always FPs. Gemini demonstrates high accuracy in individual screenshot categorization.

Detecting Inconspicuous Domain Hijacking In cases where the screenshots are similar, there may still be an attack present in the HTML's JavaScript in a way that is invisible to the user. For example, attackers might steal credentials or drop scripts into the DOM at runtime. To detect JavaScript-based non-visual attacks, we match script hashes to a PANW's malicious hash database and we also execute the samples in a JavaScript sandbox and observe their network behavior. We then cross-reference the findings from before and after the hijacking record to identify which new network behaviors the samples exhibit. Finally, we compare the list of network behaviors to a list of domains known to serve malware generated by crawling threat intelligence feeds.

Of the 58,248 cases where the screenshots are similar, we find 150 cases of the hijacked site exhibiting malicious network behavior in our sandbox. However, in 138 of those 150 cases, the malicious network behavior is present before the hijacking event, likely indicating pre-existing compromises (e.g., through vulnerable WordPress plugins) rather than hijacking-related attacks. The remaining 12 cases show network errors or missing pages before hijacking, suggesting that those sites may have undergone a change between our crawls, but again not related to hijacking attacks. While we believe performing JavaScript analysis is important to detect inconspicuous attacks, we do not have conclusive evidence that this attack vector is currently exploited in our data. Two plausible instances that we detected are discussed in §4.3.

Delayed Filtering After predicting that a record is a hijacking record, we continue to monitor its DNS traffic. Certain events in a lifetime of a domain can cause a record change to look like domain hijacking, for example, domain ownership change or hosting provider change. If we observe the hijacking record persisting over time, then it was likely an FP detection, and we reclassify it as not-hijacking. It is also possible that the hijacked domain is not recovered, the record change was caused by another type of DNS hijacking or DNS error. Our delayed filtering step can help us remove such possible FPs or hijacking records that did not recover. We additionally monitor if a hijacked domain keeps using new IP addresses in the same autonomous system or within the same service provider as the IP address in the hijacking record. If so, we mark it as not-hijacking.

3.5 Analyzing Detection

Even if our detection pipeline performs well on labeled data, the true test is its performance in production. We analyze production detections from January 1, 2024 through April 30, 2025. From the thousands of detected hijacking records, we uniformly randomly select 200 samples for manual analysis. A hundred sample from each set of *domain hijacking records* after delayed filtering and *potential domain hijacking records* before delayed filtering (see Table 2).

We categorize each sampled detection into one of five categories: *verified hijacking*, *plausible hijacking*, *good-to-block*, *low-impact FP* and *FP*. We decide by investigating the screenshots, DOMs, DNS history, archive.org [6] and the current state of the domain. We classify potential hijacking records as *FP* if the web content matches the domain owner's intended content. We distinguish *FPs* from *low-impact FPs* to understand how PANW's users are impacted. We call

an *FP* low-impact if we rarely observe the blocked DNS record and if it was blocked for a short time. We have a separate *good-to-block* category for records that are likely not domain hijacking but are part of an illicit operation (e.g., gambling, adult, illicit streaming) or clearly the result of a DNS error (the IP returns an unrelated website's content) or blocking or censorship (explicit block page). Often it cannot be definitively proved or rejected that a record is domain hijacking or not because domain hijacking operates independently of the application layer (web, email, FTP) and our visibility is limited. In case we have no evidence that contradicts or further supports hijacking, we categorize the record as *plausible hijacking*. To categorize a record as *verified hijacking*, we have three strict conditions. First, we need to observe a clear malicious intent such as defacement, phishing, malicious download, or an illicit activity (e.g., gambling or adult sites). Second, the web content or the DNS record of the victim domain should recover. Third, the hijacking attack must be temporal.

Clustering of Detections To identify domain hijacking campaigns, we use a multifaceted approach to analyze the network entities of hijacked domains and visual similarities between landing pages. We build on previous research that demonstrated the effectiveness of using *graphs* and shared relationships to detect malicious domains [21, 40]. We create our own graph model where each hijacked domain contains edges to other entities, for instance, through A and NS records from pDNS, and redirection relationships obtained by our scrapers. Additionally, we map each IP address to their corresponding ASNs, enabling us to identify clusters of hijacked domains resolving to different IPs within the same ASN, and clusters that belonged to the same ASN prior to hijacking.

As mentioned in §3.1 during domain scraping, we also capture their screenshots. We embed the screenshots with Vertex AI [16] and cluster them using DBSCAN [22] to identify visually similar landing pages. The resulting clusters are integrated into our graph model as "visual similarity" relationships, connecting source domains to cluster IDs, enhancing our existing relationship mapping.

Using all our data, we generate graphs for each potentially hijacked domain. Then, we systematically merge these graphs based on shared nodes among the domains, resulting in distinct clusters where each contains at least two domains connected through common infrastructure. We focus on merged graphs that exhibit patterns such as high concentrations of malicious nodes, differences in geolocation before and after hijacking, visual similarities between landing pages, shared hijacked IPs (or IPs within the same ASN), and lexical similarities among domain names. This approach helps us effectively identify and group related hijacking campaigns.

4 Analysis

We present an evaluation encompassing model performance, analysis of detections, verified domain hijacking detections, and potential campaign case studies.

4.1 Model Evaluation

Figure 6 compares the precision-recall curve for different types of Machine Learning (ML) classifiers. We selected the Random Forest classifier because it has the highest area under the precision-recall curve (AUC-PR). Only AdaBoost and Gradient Boosting come close

Figure 6: Precision-recall curve comparing classifiers and thresholds. Red dashed line points to a high precision (>0.99) threshold (0.52) for the Random Forest classifier. Green dotted line points to a threshold (0.16) with high recall (>0.99).

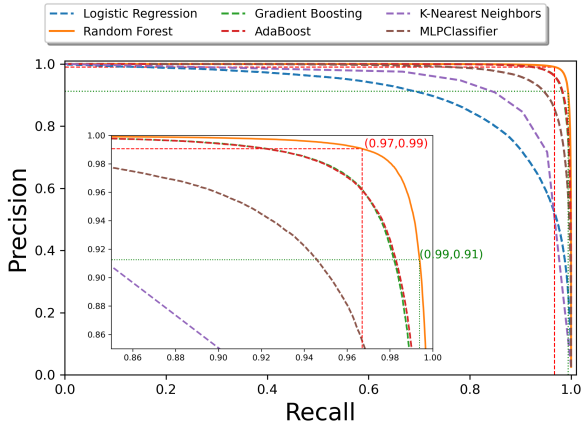


Table 7: Table summarizing our model's performance on simulated and known hijacking datasets and compares it to performance of previous work. Our work is evaluated based on root domain hijacking IP pairs. In parenthesis the values are based on hijacking campaigns. Known hijacking records are evaluated by themselves.

	Precision	Recall	F-1 Score	AUC-PR
Simulated T=0.16	0.913 (0.90)	0.994 (0.99)	0.951 (0.94)	0.997
Known T=0.16	NA	0.920 (0.90)	NA	NA
Simulated T=0.52	0.991 (0.99)	0.967 (0.96)	0.979 (0.98)	0.997
Known T=0.52	NA	0.534 (0.61)	NA	NA
Reported by [33]	0.882	0.745	0.808	0.741
ACID using [33]	0.679	0.329	0.443	0.474

in performance. In Table 7 we evaluate our model's performance using two thresholds on simulated and known hijacking datasets. Furthermore, we compare our classifier to the weighted average calculated from values reported by Houser et al. [33] and our implementation of their [33] features used in ACID. Although our known hijacking dataset spans 2014 - 2020 and Houser et al. [33] spans 2010-2020, if we exclude their data between 2010-2014, then their classifier's performance drops further (see Appendix A.5).

From Table 7, we see that our classifier has a much higher AUC-PR of 0.997 on the simulated dataset, compared to 0.4741 - 0.741 achieved by previous work [33] . We provide details of evaluating prior work in Appendix A.3. While hijacking attacks are simulated, the dataset where we inject them and the non-hijacking records come from PANW's production DNS data. We hypothesize, that the lower AUC-PR we get using features from Houser et al. compared to 0.741 reported by them [33] is because we use a more varied, larger, and complex set of benign and malicious labeled datasets as described in §3.3. Using either the results based on our reconstruction of previous work or the performance they report, ACID outperforms previous work by a wide margin.

Table 8: Number of records seen at different stages of the offline pipeline between January 1, 2024 and April 30, 2025 and the real-time pipeline between June 4, 2024 and April 30, 2025. For the real-time pipeline, new records have been filtered by the pre-candidate selection as detailed in §A.4.

Detection Stages	Record Counts (Offline)	Record Counts (Real-time)
New records	90,657,227,217	7,899,655*
Candidate domain hijacking records	1,419,247,867	5,175,059
Predicted domain hijacking records	522,387	9,232
Potential domain hijacking records	31,520	2,016
Domain hijacking records	5,121	1,252

Figure 6 depicts a crucial trade-off we have to make when deploying a model in production. First, at threshold 0.16, the classifier has a high **0.912 precision, 0.994 recall, and 0.951 F-1 score** on the simulated dataset. The model also has a high 0.92 recall on the known hijacking dataset. While compared to Houser et al. [33] we only use known hijacking records for a hold-out test and they are not used for prior training or evaluation, we still achieve a far better performance on the known hijacking dataset. Importantly, for deployment in production, we need to choose a threshold that provides higher precision than considered in previous work [33]. With threshold 0.52 our model has **0.991 precision, 0.968 recall, and 0.979 F-1 score**. The small drop in recall is well justified, as it prevents FPs and avoids causing issues for users. We also observed a drop to 0.534 in the recall on the known hijacking dataset. However, deploying a model with this threshold will still find most domain hijacking campaigns (61.2%), and it can also run automatically in a production environment every day. We are not aware of any related work that considers requirements for a production environment in a deployment where high precision is necessary.

Figure 11 in Appendix A.5, illustrates the top 10 most important features identified by our Random Forest classifier, demonstrating its effectiveness in detecting domain hijacking. The most significant indicators of hijacking are changes in NS and A records, carrying greater weight in the model's decision-making process. In Appendix A.5, we additionally evaluate how campaign types and target domains classes affect our detector's performance.

4.2 Analysis of Detections

In Table 8 we summarize the number of records we see at different stages of the offline detection pipeline during our study period. Between January 1, 2024 and April 30, 2025, we processed an estimated [1] more than 90 billion new DNS records. From these records, we selected over 1.4 billion candidate domain hijacking records, dropping the number of ML inference jobs required by nearly two orders of magnitude. Our ML model predicted 522,387 of the candidates as predicted domain hijacking records, only 0.04% of all candidates. After enrichment and analysis, we are left with only 31,520 potential domain hijacking records. After delayed filtering, we removed most of these records, keeping only 5,121 records as our final set of domain hijacking records. Overall, we end up with more than seven orders of magnitude fewer domain hijacking records compared to all the new records we observe.

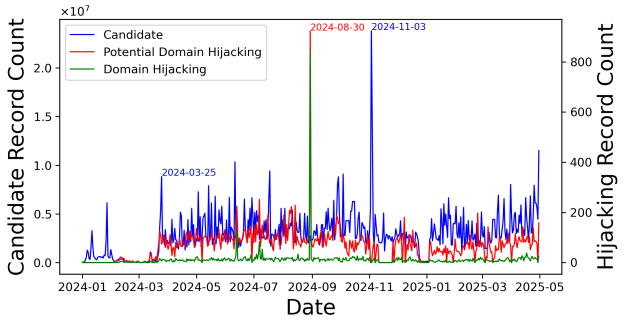


Figure 7: Record counts of detections over time during our study period.

The real-time detection pipeline has been running since June 4, 2024. The input of this pipeline are not new or unique records; instead it immediately process all records extracted from DNS response sent to customers of PANW. The real-time pipeline processed an estimated [2] 180.6 billion records between June 4, 2024 and April 30, 2025, an average of 545.5 million records per day.

In the real-time pipeline, what we call "new records" have already been filtered using multiple steps explained in Appendix §A.4 to reduce costs. Due to these early filtering methods, steps similar to the offline pipeline have reduced the number of records at each stage less drastically as shown in Table 2. All records that were processed by the real-time pipeline are also processed a bit later by the offline pipeline, thus we focus on analyzing the detections by the offline pipeline for the rest of the paper.

Figure 7 gives a longitudinal view of the detections of our pipeline, where we can see that while every day it finds millions of candidate domain hijacking records, we detect only tens of domain hijacking records. We have three notable dates. We fixed a bug in candidate selection on March 22, 2024, which caused an increase in the number of candidates and detections a couple of days later. On August 30, 2024, we found 923 potential domain hijacking records, of which 590 records were hijacked to IP addresses in China where 468 (79%) of the IP addresses belong to (AS4134). Interestingly, these ASNs were previously observed as part of China's "Anti-Fraud" censorship [68]. We also observed that these records appeared sequentially in the passive DNS data with minimal temporal differences between their initial and final occurrences. We speculate that the requests were sent in an automated fashion, potentially for testing censorship.

As domain owners can freely generate large numbers of subdomains, switching to new ISPs can cause spikes in the number of candidate records. One such spike happened on November 3, 2024, where ten root domains were responsible for nearly 18 million candidate records, and only one root domain was responsible for more than 13 million new candidate records out of the 23 million observed that day. Finally, the drops in detection in November 2024 and December 2025 were caused by issues related to the web crawling infrastructure and the passive DNS database, respectively.

[1] We do not save the exact number of new records. We saved this information only for six different weeks. We averaged these numbers and multiplied them by the number of days in our study period.

[2] Similar to new records processed by the offline pipeline we don't log the exact number of records received by the real-time pipeline.

Table 9: Gemini categories of hijacking RRs. Categories after the primary category are shown in in parenthesis.

Category Group	Record Count	Majority Categories (85+%)
No screenshot	3268	NA
No content	1315 (1484)	Error page, Under development, Default page
Normal content	357 (748)	Business website, Personal website
Interesting content	126 (264)	Login page, Webshop, Download page
Questionable content	55 (146)	Gambling, Adult

Confirming domain hijacking While we set strict rules to call a record *verified domain hijacking* (§3.5), there is still a chance that we saw a record in our pDNS dataset due to a malicious resolver, censorship or DNS error. And while some of our detections are later confirmed in the news (§4.3), to systematically reject the above non-domain hijacking causes, we analyze whether the certificates collected during crawling the predicted hijacking IP addresses are valid, as censors and malicious resolvers cannot obtain valid certificates. Differently, invalid and missing certificates do not mean that a DNS record is not a domain hijacking record.

Out of the 5,121 domain hijacking RRs, in 2,897 (56.6%) cases we have not been able to collect any certificates. Most of the missing certificates are due to the lack of web servers on hijacking IP addresses. We find two records where the certificates expired and one where the target domain was not included in the certificate. Additionally, 964 records (18.8%) are likely invalid as we only have the hashes of these certificates and we did not find them on crt.sh [18]. Finally, 1,257 (24.5%) domain hijacking records returned a valid certificate when we visited them, eliminating malicious resolvers, censorship or DNS errors as possible causes.

Manual Analysis of Samples As we described in §3.5, we uniformly randomly sampled detections to better understand the ACID's performance in production. After manually analyzing the hundred samples of domain hijacking records, we found two *low-impact FPs*, three *good-to-block*, 93 *plausible hijacking* and two *verified hijacking* records. We provide details of the verified hijacking detections and the *low-impact FPs* in Appendix A.5, where we argue that it is better to block these FPs as they point to DNS anomalies. Given our sample analysis, we expect dozens of new domain hijacking cases that could be verified from our total of 5,121 detections. In §4.3, we explore some of them in detail.

We analyzed 100 samples of potential domain hijacking records, allowing us to explore what records look like before delayed filtering. In this set, we found 78 cases of plausible hijacking records, 12 good-to-block RRs and ten low-impact FPs. Of the 10 FPs, nine were delayed filtered, minimizing their impact on PANW's customers. The one not delayed filtered was a short-lived RR with geolocation and IP string far from historical IP addresses.

We conclude that even though we block a few benign records, we successfully minimized our detector's negative impact. First, we block FP detections for only a short time. Second, in some instances, these records are related to DNS errors (e.g., *dental-artis[.]ro*). Finally, we have not received any customer complaints running ACID for more than a year. We hypothesize that the main reason for the lack of complaints is that FPs are often related to a change in hosting provider or the result of a previous DNS error.

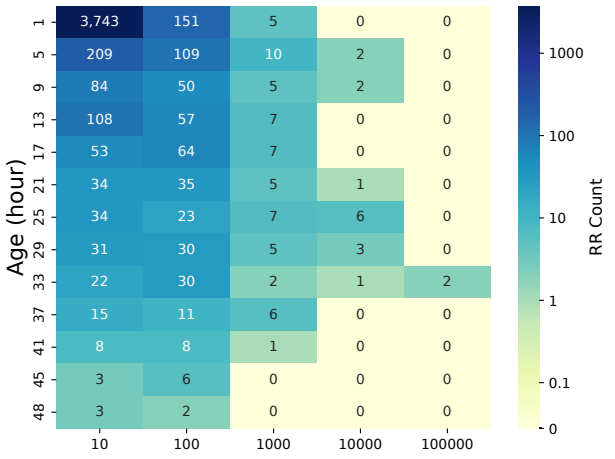
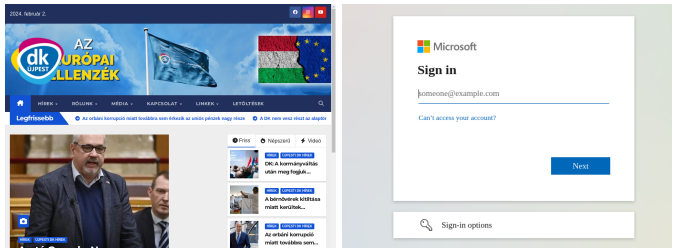


Figure 8: Age and DNS query counts of hijacking records.

Domain hijacking lifetime and query counts As expected, domain hijacking records are short-lived [2] with an average lifetime of three hours, where we define a record lifetime as the difference between the first and last time we observe it in our pDNS dataset. The standard deviation of the domain hijacking records' lifetime is 7.5 hours. In figure 8 it is apparent that most hijacking records exist for less than an hour with less than 10 queries. A notable case are the two records of *ns1/ns2.kt[.]am*, they received nearly 30,000 queries each during their 32 hours lifetime. Our crawlers received a default page and when visiting them. The domain *kt[.]am* belongs to Karabakh Telecom, which was the main ISP in the Nagorno-Karabakh region, which was seized by Azerbaijan in September 2023 resulting in the flight of all local Armenians and the end of Karabakh Telecom [37]. We see the DNS record due to a takeover or a configuration error which occurred during the aftermath of the war.

Gemini Categories Often domains do not have a functioning web server, thus in Table 9 we see that in 63.8% of the cases ACID was not able to retrieve any screenshots (thus we do not discard these records as FPs). Gemini categorized 71.0% of the screenshots as "no content," 19.5% as "normal content," and 9.8% as "questionable" or "interesting," Reviewing the 146 screenshots from the questionable category group, we find eight cases where the original content is benign and the hijacked content is gambling or adult. From twenty cases marked as phishing or scam by Gemini, we found that twelve are plausible to be malicious including two fake Microsoft download offerings, one illicit pharmacy page, two WhatsApp related scam pages, and one page offering high-paying jobs. We also found that six of them were block pages of security companies warning of phishing and we were unsure about two of them. Finally, of the three websites categorized as defaced, two were indeed defaced (*truecoding[.]in* and *ftp.conservice[.]com*) and one of them was likely not defaced, displaying only one vulgar word. Analyzing 40 samples from the 264 screenshots in the interesting category group, we found that 13 out of 23 cases Gemini confuses login and form pages with links to them. It also made two mistakes categorizing a credit and a gambling page as payment and webshop pages, respectively.



(a) Before (b) After

Figure 9: *dkujpest[.]hu* before and after hijacking.

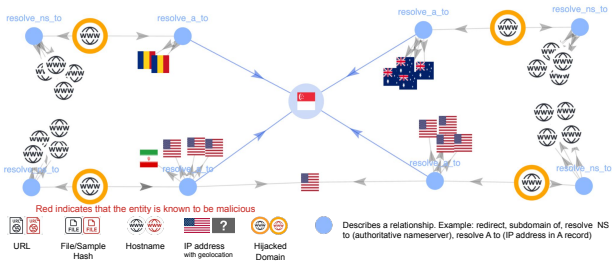


Figure 10: Merged graph depicting the gambling campaign.

4.3 Domain Hijacking in the Wild

Manually analyzing the domain hijacking detections and using techniques described in §3.5, we have identified multiple new verified domain hijacking cases, from which we explore a few here.

Domain of a Hungarian Political Party Hijacked On January 26, 2024, visitors to *dkujpest[.]hu*, the domain of the largest political opposition at the time, found themselves on the phishing page shown in Figure 9b instead of the expected page of the party (Figure 9a). The domain *dkujpest[.]hu* had a stable DNS history since 2017 using Slovakian IP addresses from the 37.9.175[.]0/24 subnet. On January 28, 2024, our offline pipeline detected that the domain resolved to the hijacking German IP address 152.70.176[.]210. The new IP address is located in a different ISP, Autonomous System, and country than the original. Two days after we detected the domain hijacking, it was covered by Hungarian news outlets [34, 67] who also made contact with the hacker.

Squarespace Domains Hijacking Campaign Our detector found several domain hijacking records for *celer[.]network* and its subdomains (*blog*, *forum*, *www*, and *cbridge*) on July 12, 2024. The root domain resolved to *185.196.9[.]153* and the subdomains to *185.196.9[.]144*, where these IP addresses were in new countries and ISPs for the targeted domains. Our system also found on the same day new NS roots for *celer[.]network*: *megan/ scott.ns.cloudflare[.]com*.

Three days after our detection, on July 15, 2024 KrebsonSecurity blogged about multiple domains affected by Squarespace domain hijacking. The main cause was **weak authentication**, where anyone who knew the email address related to a migrated domain account could take over that domain [38, 64]. Following related news [14, 17, 56, 70] we identified four related root domains (*compound[.]finance*, *compoundlabs[.]xyz*, *pendle[.]finance*, and *unstoppabledomains[.]com*). However, we did not observe suspicious DNS records for these domains with the exception of *compound[.]finance* resolving to *185.196.9[.]29* the same day as *celer[.]network*. In case of *compound[.]finance* we did not observe a new NS record, decreasing our model's confidence to a 0.41 probability, not allowing high-precision detection. Our system could detect it with a slightly lower precision threshold as discussed in §4.1.

Domains of a Large U.S. ISP and Utility Company Hijacked and Defaced In May 2024, we detected domain hijacking of two prominent U.S. companies: Cogent Communications, a Tier 1 Internet Service Provider and root server operator, and Conservice, a U.S. based utility management company. Passive DNS data reveal that the nameservers of *conservice[.]com* and *cogentco[.]com* changed to *ns[1-2].csit-host[.]com* moments before *conservice[.]com*, *ftp.conservice[.]com*, and *cogentco[.]com* resolved to one IP address 176.9.24[.]28 located in Germany. The screenshot captured from *ftp.conservice[.]com* confirmed that after the hijacking attack, the website was defaced by a group of hackers. We also noticed that this IP address was in other hacking incidents observed on *zone-h[.]org*.

Gambling Campaign We detected this hijacking campaign by identifying a merged graph—using clustering and graph-based campaign detection—in which a single IP was identified as the hijacked IP for four unique domains (*partidulnouaromanie[.]ro*, *momtaznews[.]com*, *brisbaneair.com[.]au*, *programmingwithmosh[.]com*). Notably, these domains were originally each hosted in different countries: Romania, Iran (among others), Australia, and United States respectively, pictured in the graph shown in Figure 10. The Singaporean (DIGITALOCEAN-ASN) hijacked IP hosted an Indonesian gambling webpage, as shown in Figure 14b of Appendix A.5. While most domains were hijacked for only one day according to our pDNS data, *partidulnouaromanie[.]ro* remained compromised for 34 days. All four domains have since recovered to their original content, with *momtaznews[.]com*'s original content shown in Figure 14a of Appendix A.5.

We find that the most common type of hijacking campaign websites display gambling pages. One more example is a campaign we found through our graph analysis, revealing four domains (*ettzaragoza[.]es*, *ettmurcia[.]es*, *nostalgias[.]es*, *keko[.]es*) that resolved to IP addresses within the same autonomous system (AS22612). Similarly to the previous campaign, these possibly hijacked domains displayed Indonesian-language gambling content. However, we did not find other infrastructure commonalities between these campaigns. Investigating WHOIS for this campaign reveals a potential drop-catch scenario—where domains are registered immediately after their registration expires—as WHOIS records were updated for all four domains shortly before the attack. While the domains from the previous campaign have recovered, these four domains remain in the hands of drop-catchers.

Plausible Inconspicuous Domain Hijacking We identified two instances of malware propagation among domain hijacking cases. Specifically, on May 23 2024, *lansdownassociates [.]co[.]uk* changed from an Amazon ISP in the United States (*76.223.105[.]230*) to a Hostinger IP address in the United Kingdom (*194.11.154[.]226*). A similar event occurred with *faedile[.]com* on July 2 2024, involving IP address changes from *45.63.114[.]42* to *136.244.91[.]147*. Both hijacked sites contained two additional highly malicious third-party JavaScripts which were not present before and after plausible hijacking attack. While slight visual inconsistencies existed (Appendix Figure 15), screenshots showed consistent branding, suggesting

malware distribution as the main objective. However, due to the nature of the propagated malware, we cannot definitely rule out other potential methods of exploitation besides domain hijacking.

We discuss more interesting examples in Appendix A.5 including a campaign involving 38 domains belonging to legitimate academic journals and DNS anomalies—such as DNS redirection by security vendors, domains seized by law enforcement, and DNS censorship.

5 Discussion

The ACID detection pipelines have been consistently able to detect domain hijacking while minimizing interruption to customers. Although ACID is successful, it still has limitations.

Limitations First, our pipelines are designed to detect domain hijacking of A records, as related work [33] has shown that the vast majority of domain hijacking attacks involve A record changes. Domain hijackers could modify other record types without the need to hijack A records. For example, criminals could change MX (mail server) records to point to malicious mail servers that intercept and collect emails and impersonate the owners of email addresses in the hijacked domain. Attackers could also insert CNAME records, sending traffic to their domains. Although we do not cover these record types, our methodology can be applied to other record types directly by changing the type of simulated records for training and updating the features for the ML model.

Second, an attacker can either configure a transparent MitM proxy between the victim and the original server or impersonate the original. Such setups by the hijackers would make application layer-based analysis impossible. These attacks would also come with limitations; for example, if passwords are hashed before being sent to the server, the criminals will only get the hashed password. Their modifications can be detected as soon as they send passwords without hashing. A possible trade-off to detect such attacks is to remove application layer-based analysis and filtering (e.g., analyzing screenshots and DOM files for filtering), meaning that we would rely only on DNS and geo-location data to detect domain hijacking attacks. While this approach could detect these attacks, in return, the system would cause more FPs and interruptions to customers.

Preventing Domain Hijacking While ACID helps prevent PANW's customers from visiting hijacked domains, it cannot stop domain hijacking or protect all visitors of victim domains. Registrars and DNS service providers are best positioned to prevent domain hijacking by securing their systems, requiring multi-factor authentication, and implementing strong password policies.

6 Conclusion

Domain hijacking is a rare event in the sea of hundreds of millions of new DNS records that we observe every day. ACID provides the first practical method to detect classical IPv4 record domain hijacking in real-time. Our classifier outperforms previous work by a large margin when evaluated on labeled data. To further test ACID pipelines, we have deployed them globally in production for one of the largest cybersecurity companies and have been running them for over a year. Our pipelines exhibited capabilities to detect new domain hijacking cases automatically while minimizing interruption for users, as verified by our analysis and not receiving user complaints. To enable future research on domain hijacking, we will continue to monitor domain hijacking and regularly create public reports on new attacks we discover.

Acknowledgment

The authors thank the reviewers and the shepherd for their feedback. The authors are grateful to Rebekah Houser for her input and Wanjin Li for her support with the passive DNS dataset.

References

[1] Giuseppe Aceto, Alessio Botta, Antonio Pescapè, Nick Feamster, M Faheem Awan, Tahir Ahmad, and Saad Qaisar. Monitoring internet censorship with ubica. In *Traffic Monitoring and Analysis: 7th International Workshop, TMA 2015, Barcelona, Spain, April 21-24, 2015. Proceedings 7*, pages 143–157. Springer, 2015.

[2] Gautam Akiwate, Raffaele Sommese, Mattijs Jonker, Zakir Durumeric, KC Claffy, Geoffrey M Voelker, and Stefan Savage. Retroactive identification of targeted dns infrastructure hijacking. In *Proceedings of the 22nd ACM Internet Measurement Conference*, pages 14–32, 2022.

[3] Fatemah Alharbi, Jie Chang, Yuchen Zhou, Feng Qian, Zhiyun Qian, and Nael Abu-Ghazaleh. Collaborative client-side dns cache poisoning attack. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1153–1161. IEEE, 2019.

[4] Eihal Alowaisheq, Siyuan Tang, Zhihao Wang, Fatemah Alharbi, Xiaojing Liao, and XiaoFeng Wang. Zombie awakening: Stealthy hijacking of active domains through dns hosting referral. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1307–1322, 2020.

[5] Manos Antonakakis, David Dagon, Xiapu Luo, Roberto Perdisci, Wenke Lee, and Justin Bellmor. A centralized monitoring infrastructure for improving dns security. In *Recent Advances in Intrusion Detection: 13th International Symposium, RAID 2010, Ottawa, Ontario, Canada, September 15-17, 2010. Proceedings 13*, pages 18–37. Springer, 2010.

[6] Internet archive archive.org. https://archive.org.

[7] Internet archive archive.org page for faedile.com. https://web.archive.org/web/20240601063451/https://faedile.com/.

[8] Simurgh Aryan, Homa Aryan, and J Alex Halderman. Internet censorship in iran: A first look. In *3rd USENIX Workshop on Free and Open Communications on the Internet (FOCI 13)*, 2013.

[9] Elias Athanasopoulos, Sotiris Ioannidis, and Andreas Sfakianakis. CensMon: A Web Censorship Monitor. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI 11)*, 2011.

[10] BigQuery. https://cloud.google.com/bigquery?hl=en.

[11] BigTable. https://cloud.google.com/bigtable?hl=en.

[12] Brazilian bank hijacked., 2025. https://www.darkreading.com/cyberattacks-data-breaches/cybercriminals-seized-control-of-brazilian-bank-for-5-hours.

[13] Johannes Buchner. imagehash: A python perceptual image hashing module. https://github.com/JohannesBuchner/imagehash, 2017. Accessed: April 7, 2026.

[14] Celer Network's domain hijacked. https://x.com/CelerNetwork/status/1811394743794114866.

[15] Sze Yiu Chau, Omar Chowdhury, Victor Gonsalves, Huangyi Ge, Weining Yang, Sonia Fahmy, and Ninghui Li. Adaptive deterrence of dns cache poisoning. In *Security and Privacy in Communication Networks: 14th International Conference, SecureComm 2018, Singapore, Singapore, August 8-10, 2018, Proceedings, Part II*, pages 171–191. Springer, 2018.

[16] Google Cloud. Vertex ai. https://cloud.google.com/vertex-ai.

[17] Compound Fincance's domain hijacked. https://x.com/compoundfinance/status/1811624013841727702.

[18] csrt.sh. crt.sh.

[19] David Dagon, Niels Provos, Christopher P Lee, and Wenke Lee. Corrupted dns resolution paths: The rise of a malicious resolution authority. In *NDSS*, volume 2, pages 2–3, 2008.

[20] Tianxiang Dai, Philipp Jeitner, Haya Shulman, and Michael Waidner. The hijackers guide to the galaxy:{Off-Path} taking over internet resources. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3147–3164, 2021.

[21] Fatih Deniz, Mohamed Nabeel, Ting Yu, and Issa Khalil. MANTIS: Detection of Zero-Day Malicious Domains Leveraging Low Reputed Hosting Infrastructure . In *2025 IEEE Symposium on Security and Privacy (SP)*, Los Alamitos, CA, USA, 2025. IEEE Computer Society.

[22] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[23] Farsight DNSDB passive DNS dataset. https://www.domaintools.com/products/farsight-dnsdb/.

[24] Martin Fejrskov, Jens Myrup Pedersen, and Emmanouil Vasilomanolakis. Detecting DNS hijacking by using NetFlow data. In *2022 IEEE Conference on Communications and Network Security (CNS)*, pages 273–280. IEEE, 2022.

[25] Jens Frieß, Tobias Gattermayer, Nethanel Gelernter, Haya Schulmann, and Michael Waidner. Cloudy with a chance of cyberattacks: dangling resources abuse on cloud platforms. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 1977–1994, 2024.

[26] Google. Gemini multimodal ai. https://cloud.google.com/vertex-ai/generative-ai/docs/models#gemini-models.

[27] Google. Gemini multimodal ai parameters. https://cloud.google.com/vertex-ai/generative-ai/docs/multimodal/content-generation-parameters.

[28] Google Security Blog: HTTPS by default. https://security.googleblog.com/2025/10/https-by-default.html.

[29] Shuang Hao, Alex Kantchelian, Brad Miller, Vern Paxson, and Nick Feamster. Predator: proactive recognition and elimination of domain abuse at time-of-registration. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 1568–1579, 2016.

[30] Yuanchen He, Zhenyu Zhong, Sven Krasser, and Yuchun Tang. Mining dns for malicious domain registrations. In *6th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2010)*, pages 1–6. IEEE, 2010.

[31] Amir Herzberg and Haya Shulman. Antidotes for dns poisoning by off-path adversaries. In *2012 Seventh International Conference on Availability, Reliability and Security*, pages 262–267. IEEE, 2012.

[32] Nguyen Phong Hoang, Arian Akhavan Niaki, Jakub Dalek, Jeffrey Knockel, Pellaeon Lin, Bill Marczak, Masashi Crete-Nishihata, Phillipa Gill, and Michalis Polychronakis. How Great is the Great Firewall? Measuring China's DNS Censorship. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3381–3398, 2021.

[33] Rebekah Houser, Shuai Hao, Zhou Li, Daiping Liu, Chase Cotton, and Haining Wang. A comprehensive measurement-based investigation of dns hijacking. In *2021 40th International Symposium on Reliable Distributed Systems (SRDS)*, pages 210–221. IEEE, 2021.

[34] HVG covering hijacking of dkujpest.hu. https://hvg.hu/itthon/20240130_unatkozo_tinedzser_hekkertamadas_DK_ujpesti.

[35] IP2Location IP address geolocation dataset. http://ip2location.com/database/ip2location.

[36] Andrew Kalafut and Minaxi Gupta. Pollution resilience for dns resolvers. In *2009 IEEE International Conference on Communications*, pages 1–5. IEEE, 2009.

[37] Wikipedia page on the Nagorno-Karabakh conflict. https://en.wikipedia.org/wiki/2023_Azerbaijani_offensive_in_Nagorno-Karabakh.

[38] Krebs on Security: Weak Security Defaults Enabled Squarespace Domains Hijacks. https://krebsonsecurity.com/2024/07/researchers-weak-security-defaults-enabled-squarespace-domains-hijacks/.

[39] Marc Kührer, Thomas Hupperich, Jonas Bushart, Christian Rossow, and Thorsten Holz. Going wild: Large-scale classification of open dns resolvers. In *Proceedings of the 2015 Internet Measurement Conference*, pages 355–368, 2015.

[40] Udesh Kumarasinghe, Mohamed Nabeel, and Charitha Elvitigala. Blocklist-forecast: Proactive domain blocklisting by identifying malicious hosting infrastructure. In *Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses*, pages 35–48, 2024.

[41] Victor Le Pochat, Tom Van Goetham, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation, 2019. https://tranco-list.eu/.

[42] Philip Levis. The collateral damage of internet censorship by dns injection. *ACM SIGCOMM CCR*, 42(3):10–1145, 2012.

[43] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[44] Daiping Liu, Shuai Hao, and Haining Wang. All your dns records point to us: Understanding the security threats of dangling dns records. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1414–1425, 2016.

[45] 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 *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 537–552, 2017.

[46] Graham Lowe, Patrick Winters, and Michael L Marcus. The great dns wall of china. *MS, New York University*, 21(1), 2007.

[47] Scott M Lundberg, Bala Nair, Monica S Vavilala, Mayumi Horibe, Michael J Eisses, Trevor Adams, David E Liston, Daniel King-Wai Low, Shu-Fang Newman, Jerry Kim, et al. Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nature Biomedical Engineering*, 2(10):749, 2018.

[48] Keyu Man, Zhiyun Qian, Zhongjie Wang, Xiaofeng Zheng, Youjun Huang, and Haixin Duan. Dns cache poisoning attack reloaded: Revolutions with side channels. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1337–1350, 2020.

[49] Zubair Nabi. The anatomy of web censorship in pakistan. In *3rd USENIX Workshop on Free and Open Communications on the Internet (FOCI 13)*, 2013.

[50] Frank Nagle and Daniel Yue. The latent role of open models in the ai economy. *Available at SSRN 5767103*, 2025.

[51] Hovership Nebuchadnezzar. The collateral damage of internet censorship by dns injection. *ACM SIGCOMM CCR*, 42(3):10–1145, 2012.

[52] Nick Nikiforakis, Steven Van Acker, Wannes Meert, Lieven Desmet, Frank Piessens, and Wouter Joosen. Bitsquatting: Exploiting bit-flips for fun, or profit? In *Proceedings of the 22nd international conference on World Wide Web*, pages 989–998, 2013.

[53] Gopinath Palaniappan, S Sangeetha, Balaji Rajendran, Shubham Goyal, BS Bindhumadhava, et al. Malicious domain detection using machine learning on domain name features, host-based features and web-based features. *Procedia Computer Science*, 171:654–661, 2020.

[54] KyoungSoo Park, Vivek S Pai, Larry L Peterson, and Zhe Wang. Codns: Improving dns performance and reliability via cooperative lookups. In *OSDI*, volume 4, pages 14–14, 2004.

[55] Paul Pearce, Ben Jones, Frank Li, Roya Ensafi, Nick Feamster, Nick Weaver, and Vern Paxson. Global Measurement of DNS Manipulation. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 307–323, 2017.

[56] Pendle Fincance's domain hijacked. https://x.com/pendle_fi/status/1811561786107396482.

[57] Daniel Plohmann, Khaled Yakdan, Michael Klatt, Johannes Bader, and Elmar Gerhards-Padilla. A comprehensive measurement study of domain generating malware. In *25th USENIX Security Symposium (USENIX Security 16)*, 2016.

[58] Lindsey Poole and Vivek S Pai. Confidns: Leveraging scale and history to improve dns security. In *WORLDS*, 2006.

[59] publicsuffix.org. https://publicsuffix.org/.

[60] Python-whois library. https://pypi.org/project/python-whois/.

[61] Will Scott, Thomas Anderson, Tadayoshi Kohno, and Arvind Krishnamurthy. Satellite: Joint analysis of CDNs and Network-Level interference. In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, pages 195–208, 2016.

[62] Sooel Son and Vitaly Shmatikov. The hitchhiker's guide to dns cache poisoning. In *Security and Privacy in Communication Networks: 6th International ICST Conference, SecureComm 2010, Singapore, September 7-9, 2010. Proceedings 6*, pages 466–483. Springer, 2010.

[63] Kyle Soska and Nicolas Christin. Automatically detecting vulnerable websites before they turn malicious. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 625–640, 2014.

[64] Domain Hijacking Incident Report for Squarespace. https://status.squarespace.com/incidents/cw2wf55bps15.

[65] 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 (USENIX Security 14)*, pages 191–206, 2014.

[66] Janos Szurdi, Meng Luo, Brian Kondracki, Nick Nikiforakis, and Nicolas Christin. Where are you taking me? understanding abusive traffic distribution systems. In *Proceedings of the Web Conference 2021*, pages 3613–3624, 2021.

[67] Telex covering hijacking of dkujpest.hu. https://telex.hu/techtud/2024/01/30/dkujpest-honlap-weboldal-hekkertamadas-hekkeles-defacement-adatszivargas-emailcimek-elkoveto-hekker-interju.

[68] The Tor Project. Issue #40026: Censorship analysis. Web Archive, 2025. Available at https://web.archive.org/web/20250226175344/gitlab.torproject.org/tpo/anti-censorship/censorship-analysis/-/issues/40026, archived on February 26, 2025.

[69] Martino Trevisan, Idilio Drago, Marco Mellia, and Maurizio M Munafo. Automatic detection of DNS manipulations. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 4010–4015. IEEE, 2017.

[70] Unstoppable Domain's domain hijacked. https://medium.com/@theporpoise/our-domain-was-hijacked-heres-how-blockchain-can-prevent-this-from-happening-to-you-9b9696d80d07.

[71] Virus Total. https://www.virustotal.com/gui/home/upload.

[72] Thomas Vissers, Timothy Barron, Tom Van Goethem, Wouter Joosen, and Nick Nikiforakis. The wolf of name street: Hijacking domains through their nameservers. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 957–970, 2017.

[73] Nicholas Weaver, Christian Kreibich, and Vern Paxson. Redirecting DNS for Ads and Profit. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI 11)*, 2011.

[74] Duane Wessels. Dns cache poisoners lazy, stupid, or evil? *NANOG. NANOG*, 2006.

[75] WhoisXML WHOIS databse. https://whois.whoisxmlapi.com/overview.

[76] Wikipedia. Potentially unwanted program. https://en.wikipedia.org/wiki/Potentially_unwanted_program.

[77] Bin Yu, Daniel L Gray, Jie Pan, Martine De Cock, and Anderson CA Nascimento. Inline dga detection with deep networks. In *2017 IEEE international conference on data mining workshops (ICDMW)*, pages 683–692. IEEE, 2017.

[78] Lihua Yuan, Chao-Chih Chen, Prasant Mohapatra, Chen-Nee Chuah, and Krishna Kant. A proxy view of quality of domain name service, poisoning attacks and survival strategies. *ACM Transactions on Internet Technology (TOIT)*, 12(3):1–26,

2013.
[79] Lihua Yuan, Krishna Kant, Prasant Mohapatra, and Chen-Nee Chuah. Dox: A peer-to-peer antidote for dns cache poisoning attacks. In *2006 IEEE international conference on communications*, volume 5, pages 2345–2350. IEEE, 2006.
[80] Fenglu Zhang, Yunyi Zhang, Baojun Liu, Eihal Alowaisheq, Lingyun Ying, Xiang Li, Zaifeng Zhang, Ying Liu, Haixin Duan, and Min Zhang. Wolf in sheep's clothing: Evaluating security risks of the undelegated record on dns hosting services. In *Proceedings of the 2023 ACM on Internet Measurement Conference*, pages 188–197, 2023.
[81] Yunyi Zhang, Baojun Liu, Haixin Duan, Min Zhang, Xiang Li, Fan Shi, Chengxi Xu, and Eihal Alowaisheq. Rethinking the Security Threats of Stale DNS Glue Records. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 1261–1277, 2024.
[82] Yunyi Zhang, Mingming Zhang, Baojun Liu, Zhan Liu, Jia Zhang, Haixin Duan, Min Zhang, Fan Shi, and Chengxi Xu. Cross the Zone: Toward a Covert Domain Hijacking via Shared DNS Infrastructure. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 5751–5768, 2024.
[83] Xiaofeng Zheng, Chaoyi Lu, Jian Peng, Qiushi Yang, Dongjie Zhou, Baojun Liu, Keyu Man, Shuang Hao, Haixin Duan, and Zhiyun Qian. Poison over troubled forwarders: A cache poisoning attack targeting DNS forwarding devices. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 577–593, 2020.

A Appendix

A.1 Ethical Considerations

Our dataset contains query type, queried FQDN, and rrdata in responses collected on firewalls. We also crawl the information of the queried FQDN, such as website screenshots, domain Whois, IP geolocation, and domain certificate. None of the data is considered personally identifiable information (PII), and we only analyze and process DNS data that are covered by user agreements that permit analysis. Our dataset is processed and analyzed on secure internal servers. In addition, the organizations where the firewalls are deployed are aware of the block action and may configure the actions to take for hijacking (e.g., block/alert/allow). In the case of FPs, they can add the blocked records to their local trust list or/and submit FP reports to us. Finally, we have sent notification emails to the owners of **all** hijacked domains, in some cases receiving confirmation of the hijacking.

A.2 Methodology

Simulation Target Domain Classes To select a large variety of target domains, we vary the complexity of pDNS and geolocation history for each as shown in Table 10. Additionally, each domain has a 50% chance to be owned by customers, to use self-hosted nameservers, or to have a TLD that matches the IP country code.

Simulation Campaign Types The main parameters for creating simulated domain hijacking campaigns are listed in Table 11. From small to large, we increase the number of target domains, IP addresses, and name servers used. We maximize the number of IP addresses and name servers per target domain that an attacker can use, as we have never seen a case where more than these amounts were used.

Table 10: The number of IP CCs and NS roots per target domain class.

Class	# of IP CCs	# of NS roots
Easy	1	1
Medium	2-12	2
Hard	13 - 70	2-12

Table 11: The number of target domains, IP addresses and name servers used for each type of simulated domain hijacking campaign.

Campaign Type	# Target Domains	# IP address	# Name Servers	Max IP/Domain	Max NS/Domain
Small1	1	1	1	1	1
Small2	1	4	2	2	2
Medium1	4	3	2	2	2
Medium2	8	10	2	2	2
Large1	20	4	2	2	2
Large2	40	32	4	2	2

Table 12: Percent of hijacking records where ISP or ASN is present in the hijacked domain's history.

Dataset	Total	Same ASN	Same ASN %	Same ISP	Same ISP %	Both Same	Both Same %
Simulation	45,216	1,539	3.40%	869	1.92%	699	1.55%
Known	88	1	1.14%	0	0.00%	0	0.00%

Table 13: The capability of ACID to detect hijacking records for simulated records where the ISP or the ASN is present in the hijacked domain's history.

Segment	Total pairs	Detected (t=0.50)	Det% (t=0.50)	Detected (t=0.5175)	Det% (t=0.5175)
Same ASN	1,539	1,333	86.61%	1,319	85.71%
Same ISP	869	703	80.90%	696	80.09%
Both same	699	547	78.25%	540	77.25%

Reuse of ASN/ISP In Table 12, we can see that among the simulated attacker IP addresses, 3.4% have ASNs that match the historical ASNs of target domains, and 1.92% have ISPs that match the historical ISPs. Table 13 shows that ACID is able to detect 77.25%-85.71% of cases with a matching ASN or ISP. Only in one known hijacking case did the attacker use an IP address matching historical ASNs; our SHAP value analysis indicates that features unrelated to ASNs/ISPs (e.g., the top three features shown in Figure 11) were the reason this detection was missed.

Although detecting domain hijacking is harder when an attacker uses IP addresses with ASNs/ISPs that match the target domain's history, overall, the results demonstrate ACID's ability to detect domain hijacking even in these situations.

Further Details of Detection Design While the prefiltering step removes all non-A records from consideration for detection, all record types are kept to calculate features such as "Number of RR types in new RRs". ACID uses delayed filtering 48 hours after initial detection, providing a sweet spot for our customers between defensive coverage and minimizing FP impact. On the one hand, domain hijacking attacks persist for only a short time, so this threshold ensures protection. On the other hand, legitimate migrations often retain previous records for a period, minimizing the impact on our customers in the event of FP detections. For feature calculation, we use the past 400 days of pDNS data. Records older than 7 days are considered historical, and those newer than 7 days are considered new records. If WHOIS data indicates an ownership change in the past 30 days, then we do not consider candidate hijacking records as domain hijacking.

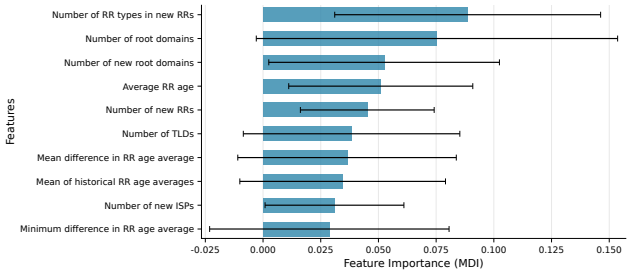
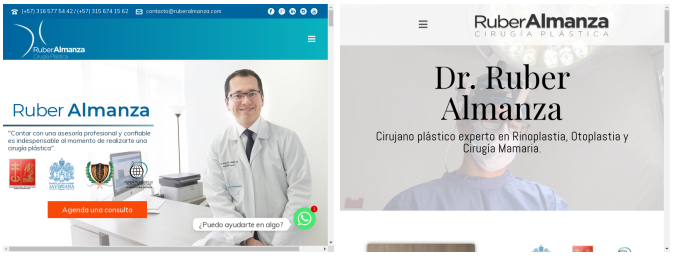


Figure 11: Top 10 feature importance scores from the Random Forest classifier ranked by Mean Decrease in Impurity (MDI)



(a) Before **(b) After**

Figure 12: The domain *ruberalmanza[.]com* before and after changing hosting providers.

Using Geminig 2.5 Flash introduces a latency of 0.93-14.74 seconds (average 1.85s), which is negligible relative to the offline pipeline's hours-long processing time and the real-time pipeline's approximately 10 minutes. For both pipelines, the main contributors to time-to-detection are pDNS feature calculation and web crawling.

Example Hosting Provider Change After six years, the domain ruberalmanza[.]com changed hosting providers. We observed the new IP address 92.112.191[.]81 on September 2, 2024. While the two screenshots shown in Figure 12 are clearly created by the domain owner, they look very different possible due to an update to the web content itself. Gemini can handle such cases with very high accuracy, while using hashes of the screenshots would miss that both contents were created by the domain owner.

A.3 Model Evaluation

To enable a comparative performance analysis of our features against those proposed by Houser et al. [33], we implemented the features described by the authors and evaluated them on our dataset. This comparative analysis utilized the same classifier and parameters specified by the authors, and the comparisons were performed on the data detailed in §3.3 but only using new RRs observed on 2023-09-07. However, since [33]'s features are calculated at the root level, not the record level, it leads to instances where samples shared the same root domain but had different labels. To have a fair comparison, we conducted two sets of experiments. In the first experiment, we used [33]'s features without modification. In the second experiment, we adjusted the labels of all samples belonging to a given root domain to "hijacking" if any sample within that root domain was labeled as such. Table 14 presents the results of these experiments.

Table 14: Domain hijacking detection: ADHD vs. ADHD using [33].

Experiment	Precision	Recall	F1-Score	AUC-PR
ADHD	0.99	0.9835	0.9867	0.9989
ADHD using [33]	0.6795	0.3294	0.4431	0.4741
ADHD using [33] (label adj.)	0.6866	0.2977	0.4147	0.4409

A.4 Real-time Pipeline Design

The real-time domain hijacking detection pipeline (Figure 13) consists of four modules. First, records undergo a real-time prefiltering module that reduces the volume to be processed. Second, records are batched for more cost-effective feature calculation. Third, they go to the real-time detection module. Finally, the offline precalculation module which enhances the real-time detection module's speed and efficiency.

Real-time Prefiltering Module Our prefiltering module instantly processes incoming DNS records from customers of PANW, following engineering best practices to reduce dataset size early for efficient processing. The prefiltering is the fastest module in the real-time pipeline by a large margin as it runs in the order of seconds.

Unlike the offline pipeline, we need to process all the records which makes prefiltering crucial for cost-efficient operation. We filter popular DNS records that appear thousands of times per day in our pDNS datasets and have been seen for more than four days. In practice, popular domains typically recover from hijacking much quicker than four days, making this filtering method effective for reducing processing volume without excluding hijacking incidents.

We filter out domains with a large number of diverse set of records across multiple ISPs, ASNs, and countries, as their frequent record changes make hijacking detection unreliable. While we do not filter these domains in the offline pipeline, in the real-time pipeline they help us to save time and cost.

Unlike the offline detector, we need to check whether a record is newly observed, an RR that we have never seen at the time of checking. pDNS API allows us to keep only new records and filter records where the rrnames are NOHs (here we use a 90 day threshold). Additionally, pDNS API allows a lightweight candidate selection, where our pipeline analyzes if the /24 subnet of the new A record is in the set of historical /24 subnets of the rrname. The difference compared to candidate selection is that we do not perform this for all /24 subnets of the root domain and its subdomains.

Real-time Batching Module As candidate selection and feature extraction require large datasets, it is computationally much cheaper if we batch the DNS records we need to analyze. With limited computational capacity, batching also makes processing faster as these operations have approximately the same runtime for small batches (e.g., dozens of records) or just for an individual record. The batching module waits for a predefined period of time (e.g., one minute) to collect records to process, then it sends the collected record batches to the Real-time Detection Module.

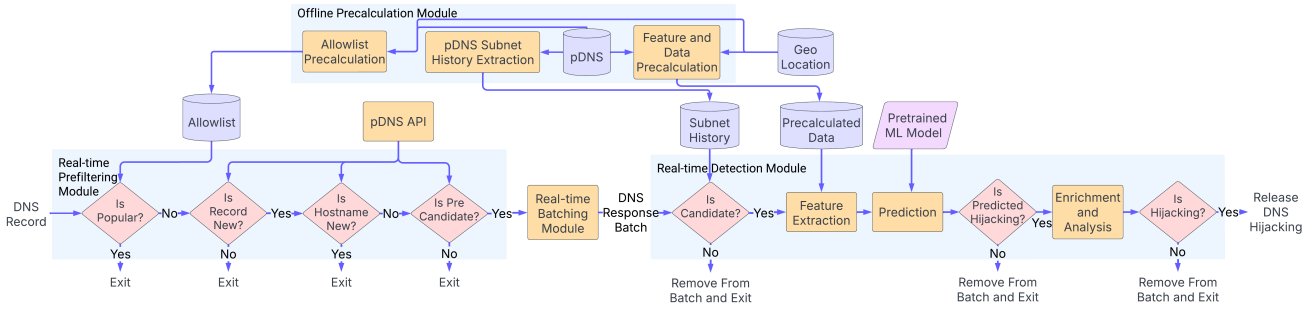


Figure 13: Design of the real-time domain hijacking detection pipeline.

Offline Precalculation Module The Real-time Prefiltering and Batching Modules are lightweight and run in seconds. However, the Real-time Detection Module runs in the order of hours and is expensive without the Offline Precalculation Module as it needs to process all pDNS history of records received from the batching module. This module precalculates pDNS data once per day for real-time candidate selection and feature extraction. Preprocessing consists of filtering by rrtype, removing invalid records, removing unnecessary fields, precalculating new fields necessary (e.g., subnets of IPs), and partitioning data for faster search by root domain. Overall, it minimizes data and makes it ready for the specific tasks required by the pipeline.

Real-time Detection Module The Real-time Detection Module works very similarly to the offline detection pipeline. The difference is that it needs to process less records due to the Real-time Prefiltering Module and it runs faster due to the Offline Precalculation Module. It consists of five main steps.

First, the batch of records undergoes a full candidate selection as described in §3.3. If it finds a subnet in the root's history, it will ignore the related new records. Otherwise, it will consider them as candidate domain hijacking records and send them to feature extraction. Second, feature extraction will calculate features for all candidate domain hijacking records leveraging the precalculated data for efficiency. Third, the pretrained ML model use these records as input and returns their likelihood of being domain hijacking. If the likelihood is larger than a predefined threshold, it considers the candidate as a predicted domain hijacking record. Fourth, it uses the enrichment and analysis module, similar to the offline pipeline, to output domain hijacking records. Note that delayed filter is used for both the offline and real-time detection pipelines.

A.5 Analysis

Known Hijacking For comparing known hijacking collected by Houser et al. [33] to cases collected by us, we use all the domains reported by them. In total, they found 86 victim domains, but they have data only for 68 of them. Since they don't clarify which domains they have data for, we consider all of them. We found 45 victim domains that we have data for. Out of the 86 victim domains only 38 were after 2014, and since our pDNS start in 2014 we consider these 38 domains. The overlap is 28 domains, Houser et al. has 10 domain we do not, and we have 17 domains they do not have. Overall the two datasets are not equal they are comparable between the dates 2014 and 2020.

Table 15: Table provides details on our model's recall on simulated hijacking records per campaign type, target domains class, and target domain category.

Campaign Type	Recall
Small1	0.914
Small2	0.915
Medium1	0.958
Medium2	0.961
Large1	0.968
Large2	0.975

Target Domain Class	Recall
Easy	0.969
Medium	0.923
Hard	0.942

Target Domain Category	Recall
Web-based email	0.971
Cryptocurrency	0.852
Military	0.946
Government	0.963
Financial services	0.963
PANW	0.953
Tranco popular	0.942
Customer queried	0.960
Other	0.945
Customer owned	0.980

Effect of campaign type and target domains on detector performance Table 15 shows how different campaign types and target domain categories and classes affect our detector's performance. We find that our model more effectively detects larger campaigns affecting multiple domains, as these campaigns generate abnormal values across a greater number of features. For example, hijacking A records will be new for more domains, and we will also see more new record types. We do not observe significant differences across target domain categories, except cryptocurrency domains being harder to detect. Our SHAP [47] analysis of the misclassified records reveals that the absence of nameserver change (randomly set in our data) was the most significant factor contributing to the misclassifications of such samples.

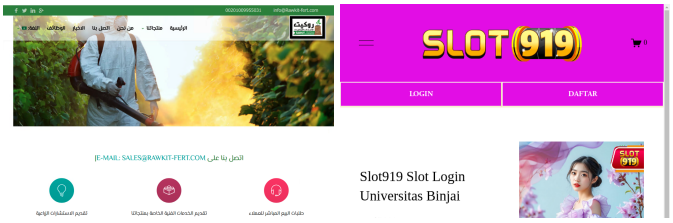
Manual Analysis of TPs The two true positive detections we found during the manual analysis in §4.3 were for rawkit-fert[.]com and radheyprecisiontools[.]com. The domain of a fertilizer company, rawkit-fert[.]com, since 2019 has used IPs located in the US and Germany. On 3 January 2025 we detected that it pointed briefly to 46.250.231[.]75. The hijacking IP address returned a page related to gambling as shown in Figure 16b. Since then, rawkit-fert[.]com has recovered. The domain of a construction company, radheyprecisiontools[.]com, has used a German IP address 89.117.53[.]211 since 2021, but it has briefly resolved to 103.93.94.106, an Indian IP address on August 18, 2024. This IP address returned a shady download website that likely distributed PUPs (potentially unwanted programs)

(a) Before (b) After

Figure 14: *momtaznews[.]com* before and after hijacking.

(a) Before (b) After

Figure 15: *faedile[.]com* before [7] and after domain hijacking.



(a) Before (b) After

Figure 16: *rawkit-fert[.]com* before and after domain hijacking.

[76]. Our DNS data indicates that the domain has recovered, but we currently get a suspended error message.

Manual Analysis of FPs Both low-impact FPs were curious cases as the records were short-lived and the detected hijacking IP addresses were part of different ISPs and ASNs compared to the historical IP addresses or the IP addresses after the hijacking. One of them, the website of a Romanian dentist office (*dental-artis[.]ro*), was an interesting case, as the most recent IP before the detected record pointed to the website of a Hungarian construction company (*2pgranit[.]hu*) for more than two months (between June 6, 2024 and July 8, 2024). Although WHOIS indicates that the domain was registered on May 6, 2024, we see on archive.org that the domain used to have the same website before the registration date as it has now. We hypothesize that the domain expired and was re-registered, and our detector found an event when an initial DNS configuration error was first corrected. We argue that it is better that we detected these low-impact FPs as they point to DNS anomalies.

Figure 17 is the screenshot of the domain hijacking of conservice[.]com from §4.3. We provide an online appendix at https://janos.szurdi.com/content/academicpapers/domain-hijacking-online-appendix.pdf that discusses further interesting findings.

Figure 17: Domain ftp.conservice[.]com defaced.

A.6 Prompt Design

We use two main prompts for screenshot similarity comparison and detecting website categories. The categories were Empty page, Error page, Loading page, Short text only, Verify human or Captcha, Parked, Under development, Default page, Business website, Personal website, Other website with content, Other website without content, Webshop, Login page, Download page, Streaming, Gambling, Adult, Torrent, Defaced webpage, Payment page, Form page, Scam page, and Phishing page. Although we use Gemini 2.5 flash, current open-source models provide viable alternatives, as they already achieve performance similar to closed-source models [43, 50].

Similarity Comparison Prompt:

> *First, tell me if the topic of the two images of the websites are similar, answer with "yes" or "no". Second, tell me if they contain the same language, answer with "yes" or "no". Use the format* `{"is_similar":"response","is_same_language":"response"}`.

Category Classification Prompt:

> *List at most the top four categories that apply to the image of the website from these categories (see Section A.6). Only use categories "Other website with content" and "Other website without content" if none of the other categories apply. The categories "Business website" and "Personal website" are mutually exclusive. Use only the one that applies best from "Under development" and "Default page" categories. Use the format* `{"categories":["cat1","cat2"]}`.