C-FRAME: Characterizing and measuring in-the-wild CAPTCHA attacks

Hoang Dai Nguyen[§]
Louisiana State University
hngu281@lsu.edu

Karthika Subramani
Georgia Institute of Technology
ksubramani@gatech.edu

Bhupendra Acharya[§]
CISPA
bhupendra.acharya@cispa.de

Roberto Perdisci
University of Georgia
Georgia Institute of Technology
perdisci@uga.edu

Phani Vadrevu[§]
Louisiana State University
kvadrevu@lsu.edu

***Abstract*—In this paper, we design and implement C-FRAME, the first measurement system to collect real-time, in-the-wild data on modern CAPTCHA attacks. For this, we study the recent evolution in the protocols of CAPTCHAs as well as human-driven farms that facilitate attacks against CAPTCHAs. This study leads us directly to the discovery of a unique vantage point to conduct a global-scale CAPTCHA attack measurement study. Harnessing this, we design and build C-FRAME to be CAPTCHA-agnostic and ethically considerate. We then deploy our system for a 92-day period resulting in capturing of 425,257 CAPTCHA attacks on 1417 sites.**

In order to characterize these attacks, we leverage a carefully designed qualitative analysis approach using 3 analysts. Our study results in delineation of 34 different CAPTCHA-attack categories with several interesting real world attack examples. Twitter received the largest number of CAPTCHA attacks overall (about 255,480 attack requests) most of which attempt to create bot accounts. We also categorized and captured attacks such as ticket scalping attempts (e.g. a Taylor Swift concert event in Brazil), fraudulent lawsuit claims, and abusive appointment booking attempts (e.g. a Spain visa site in China). We also found CAPTCHA-assisted attempts to download data from government website (e.g. websites of 20 US states). We ascribe our attacks to 58 different countries across 5 continents. We present a detailed measurement analysis to give insights on this attack data and also suggest some future potential remediation measures that can be inspired by our system.

1. Introduction

CAPTCHAs are web-abuse prevention tools that have been in existence for more than two decades [1]. Unfortunately, CAPTCHA implementations have long been riddled with usability issues [2]–[4]. Because of this, CAPTCHA services have begun to embed measures to decrease the "trigger rate" of their CAPTCHA puzzles [5] in order to mitigate user annoyance. As a result, regular users might underestimate the actual scale of deployment of CAPTCHAs on the internet. In fact, a recent study which attempted to measure the presence of CAPTCHAs solely on account-creation pages of top 200 websites showed that at least 60% of them have deployed CAPTCHAs [4].

Despite such widespread and long-term deployment, to our knowledge, no study has yet performed a global-scale measurement analysis of web sites whose CAPTCHA implementations are being targeted for attacks. But, the lack of such a study is understandable. Unfortunately, academic researchers have hitherto not had access to a global-scale vantage point to conduct such a study. On the other hand, industry-based researchers who work with CAPTCHA service companies might have access to a large-scale CAPTCHA attack data identified by in-house abuse detection methodologies. However, such researchers also have two important road blocks. First, their data is only restricted to the client sites associated with their own CAPTCHA service and excludes all other services. But, second and probably more important, publishing research on attacks on their own clients' sites stands in direct conflict with the business interests of their companies. This might explain why we have not yet had any such public study on characterizing and measuring in-the-wild attacks on CAPTCHAs. But, the lack of such knowledge, nevertheless, represents an important knowledge gap for the security community.

In this paper, we fill this gap. Our work is based on a couple of key observations. First, we notice that CAPTCHA implementations have undergone significant changes in recent years with the landscape shifting from text to modern *behavioral* implementations. We do this by closely studying the protocols of six modern CAPTCHA APIs (Section 2). Second, we observe that "CAPTCHA-farms" which facilitate human-driven CAPTCHA abuse have also accordingly adapted to this landscape shift since their last systematic study in 2010 [6]. We infer this by closely studying the APIs of six different CAPTCHA-farms (Section 3). In general, in both these shifts there is an increase in complexity

§. Part of this study was conducted when these authors were at the University of New Orleans.

of the protocols that both the CAPTCHA services and the CAPTCHA-farms are using. Incidentally, this resulted in a final key observation that we were able to make. We noticed that an inherent part of this complexity increase (across *all* CAPTCHA services and farms we studied) is that *fine-grained information about target sites of the* CAPTCHA *attacks should now be necessarily transported to the human workers in order to carry out the attacks*. Specifically, we observed that CAPTCHA-farms that attempt to accommodate modern behavioral CAPTCHAs are forced to divulge the URL of the attacked sites to the human workers who ultimately solve the CAPTCHA challenges. This presents an invaluable vantage point to measure real-world attacks on modern CAPTCHA implementations.

We then proceeded to leverage this vantage point by designing a global CAPTCHA attack measurement system named C-FRAME (CAPTCHA-Farm Reconaissance and Abuse Measurement System). Our designed system emulates human workers of modern CAPTCHA-farms and collects information about URLs that are being targeted by CAPTCHA attackers. We designed this system to be able to conduct large-scale measurements while being CAPTCHA-service agnostic and ethically considerate (Section 4). We deploy this designed system over a 92-day period resulting in capturing of 425,257 CAPTCHA attacks on 1417 sites. We then utilize a unique hybrid approach of Qualitative (Section 5) and Quantitative analysis to characterize and then measure the attack dataset that we collected.

In summary, we make the following contributions:

1) We systematize the working protocols of modern behavioral CAPTCHAs and their human-driven farms leading to multiple insights. Among them, a key insight was that the operation of modern CAPTCHA-farms unavoidably divulges information about the targeted sites to the farm worker software.
2) Capitalizing on the above finding, we designed and implemented an ethical, CAPTCHA-service agnostic measurement system named C-FRAME that simulates farm workers. We deployed it over a 90-day period on two CAPTCHA-farm systems yielding data on 425K in-the-wild CAPTCHA attacks on 1,417 sites.
3) We designed and implemented a 3-phased Grounded-theory based pipeline for Qualitative analysis of targeted URLs on a subset of collected sites (602 sites). It revealed a new taxonomy of CAPTCHA attacks spanning 34 attack categories and 6 meta categories across several popular sites and real-world events
4) We also perform measurement analysis which revealed wide geographical nature of CAPTCHA attacks, targeting more than 12 different services across 58 countries and 5 continents. The analysis also confirmed the stability of the 34 attack categories.

2. Understanding Behavioral CAPTCHAS

CAPTCHAs are a type of challenge-response test used to determine whether the user is human or a bot. Text CAPTCHAs which challenge users via visual or auditory puzzles were very prevalent in the past. However, over the past few years several machine learning-based attacks have been proposed against these text CAPTCHAs that resulted in their decline [7]–[15] . In fact, a study conducted in 2023 [4] has shown that less than 12% of the top CAPTCHA-laden websites use text CAPTCHAs.

In 2013-14, Google took a stark departure from text CAPTCHAs [5], [16] by announcing what we term as "**behavioral CAPTCHAS**" in this paper. A key cornerstone of this type of CAPTCHA is the intent of the CAPTCHA services to actively consider the user's behavior as an important signal in deciding if the visitor is a bot. In order to achieve this, the JavaScript source code provided by the CAPTCHA services will run as a first-party in the client's website and thus have the ability to track all user actions. Such tracking ability allows the CAPTCHA services to collect the behavior of users on the web page (such as mouse movement patterns etc.) and harness this to power intelligent behavior-based bot detection systems. These behavioral CAPTCHAs can broadly be classified into two kinds as described below.

Puzzle-based CAPTCHAS. Behavioral CAPTCHAs combined with puzzles have been very popular across various providers since Google first announced them in 2013. The puzzles are very dynamic with interactive elements throughout instead of the static puzzles used previously in text CAPTCHAs. For example, one of the puzzles first proposed by Google presented the users with a 3x3 grid of images and required the users to click/tap on all images that present an object of interest [5]. This puzzle is still prevalent in many CAPTCHA implementations. Some other examples of puzzles are Arkose Labs' orientation/rotation puzzles and GeeTest's slider puzzles [4]. Note that apart from their core functionality of challenging the visitors, these dynamic puzzles are all designed to elicit behavioral information from the users in the process of solving the puzzles.

Puzzle-less CAPTCHAS. Given that all behavioral CAPTCHAs already capture signals that expose bot presence, some implementations keep the presence of explicit puzzles optional. For example, Google's original 2013 proposal [5] (current name: "reCAPTCHA v2") asks the user to first click on a checkbox that simply says, "I'm not a robot". As in most cases this yields sufficient signal to make the bot/not judgement, Google marketed this as a pre-filtering stage that enhances the usability for many legitimate users by avoiding painful puzzles. The recent advances in machine learning allowed for products such as Google's reCAPTCHA v3 (2018) and Cloudflare's Turnstile (2022) which are completely devoid of CAPTCHA puzzles and solely rely on the behavioral signals to compute a confidence score for the user. In these products, the score is computed at the onset of a site-defined event such as, the submission of a HTML form. It is important to note that this avoidance of puzzles (which is arguably the most recognizable trait of CAPTCHAs) allowed marketing teams of entities such as Cloudflare to market their products as "CAPTCHA alternatives" [17]. However, as we will see in the rest of the section, these "alternatives" follow similar implementation protocols as other behavioral CAPTCHAs. Further, this implies that

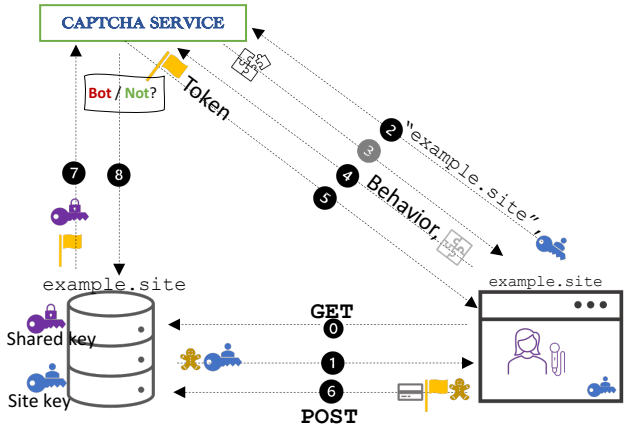

Figure 1: Working protocol of a behavioral CAPTCHA.

these puzzle-less CAPTCHAs are subject to the same kind of abuses by human-driven CAPTCHA farms as the ones with puzzles. We will see evidence for this in terms of our measurements in Section 7.

A recent measurement study showed, 88% of CAPTCHAs on top websites are behavioral CAPTCHAs [4]. Thus, we focused solely on these behavioral CAPTCHAs alone (both with and without puzzles) in this project[1].

2.1. Behavioral CAPTCHA Workflow.

In this section, we attempt to systematize the technical workflow of behavioral CAPTCHAs. To the best of our knowledge, *ours is the first work to do this*. For this, we studied the API documentation of all the following six CAPTCHA services: Google reCAPTCHA v2 and v3, hCaptcha, Arkose Labs, GeeTest, and Cloudflare Turnstile which cover all the behavioral CAPTCHAs seen in top sites [4]. Interestingly, despite stark differences in marketing strategies between various CAPTCHA services, we saw that all of their client sites are required to follow the same protocol for utilizing their services. We discuss this protocol below.

Fig. 1 shows a pictorial summary of our findings systematizing the working of all six CAPTCHA implementations we considered. For this discussion, we consider an imaginary website: `example.site` that needs to deploy a CAPTCHA in order to protect their service, say, a concert ticket sales page, from bot attacks. For this, the developers of `example.site` should register themselves with a CAPTCHA service provider and obtain two sets of keys: (1) a secret *shared-key* that is used for communication with the provider and, (2) a *site-key* that is used to uniquely identify their site. The developers then embed a snippet of JavaScript code provided by the CAPTCHA service as a first-party in their sales page. This code snippet includes an explicit mention of the site-key that has been given to the developers. This code will capture the behavior of the users on the page as discussed before. For this reason, the CAPTCHA providers dissuade the site developers from "lazy loading" of the CAPTCHA script and instead recommend browser features such as `pre-connect` resource hints to establish early connections and maximize user data collection [18].

Now, consider a legitimate user who visits `example.site` for purchasing a ticket. Their `GET` request ❶ results in a cookie being set in their browser and the CAPTCHA script (along with site-key) being loaded in their browser ❶. The script then makes a query to the CAPTCHA service in which the site-key is included explicitly ❷. Often, the domain name of the site is also included in the HTTP request as part of the `Referer` header. However, since the referrer header can be suppressed by the users, the CAPTCHA services ultimately rely on the site-key to infer the client site that is associated with this request. The CAPTCHA providers respond with a puzzle to be rendered to the user ❸. Note that this step is optional as some behavioral CAPTCHAs do not carry an explicit puzzle as discussed earlier[2]. Next, whenever the user solves the rendered puzzle (or, in the case of puzzle-less CAPTCHAs, when the user triggers a form submission action), the client page sends an encoded summary of users' behavioral signals thus far and (optionally,) a solution of the puzzle to the service ❹. The CAPTCHA service then takes these two into account and computes a *risk score* which can be uniquely identified by a *token* string. This token which will ultimately help the service characterize the user in question is sent back to the client site ❺.

The client-side code of `example.site` web server can then attach the received token to form data that is being sent to the server. For a ticket sales page that we are considering in this running example, this could include information such as event data, credit card information as well as cookie data ❻. All this data is typically sent in a `POST` request to the server side. The web server will then send the token along with the secret shared-key to the CAPTCHA service to receive the "risk score" associated with this user session directly from the CAPTCHA service ❼. Note that this token-based design is essential for CAPTCHA implementations as directly receiving the risk score from the clients is prone to client-side manipulations by abusive users. It is also important to note that these tokens typically have a service-specific expiration timeline. Some services such as Google are explicit about this by stating a clear expiration time-period of two minutes from the time of issue for all their CAPTCHA tokens. On the other hand, other services such as hCaptcha are vague about this with documentation that simply states that the tokens should be "verified within a short period of time after being issued".

Finally, once the token is sent the CAPTCHA service responds with a decision about whether the user session in question represents a bot or not ❽. In cases where there is an explicit CAPTCHA puzzle, the output from the captcha service is typically a binary flag. However, in the case of puzzle-less CAPTCHAs, services might resort to providing a

1. In the remainder of this paper, we sometimes use the term "CAPTCHA" to refer to "behavioral CAPTCHAs" for brevity.

2. Step ❸ is in gray color in Fig. 1 to indicate this optional nature.

risk score that merely represents the chance that the visit is a bot. The decision of what actions are to be taken (based on what thresholds) are left to the site developers themselves.

3. Understanding CAPTCHA Farms

Armed with an understanding of how behavioral CAPTCHAs work, we are now in a position to discuss the technical details of how they are compromised by human-driven CAPTCHA farms. We first provide a brief background about how these farms have operated in the past. We will then provide details about how the farms have now evolved their operations with the advent of behavioral CAPTCHAs. Finally, we will discuss implications of these operational insights for our paper as well as future research.

Background. CAPTCHA-farms have been in existence for more than a decade now. They leverage the cheap labor in developing countries as workers to help in conducting attacks against bot prevention solutions deployed by websites. In 2010, Motoyama et al. have studied the economies of these CAPTCHA-farms [6] during a time in which text CAPTCHAs were prevalent. Their work showed that CAPTCHA farms operate two platforms: (1) for their customers (i.e. web attackers) to submit their CAPTCHA attack requests via paid web APIs (*solver APIs*); (2) for their workers to solve CAPTCHAs from the attackers and earn money via desktop-based software or farm web sites (*worker platforms*). The APIs receive static CAPTCHA puzzles (such as image/audio files) which is distributed by the farms to the software being used by the workers. We find that this model has persisted to this day. In fact, some solver API platforms such as "Anti-Captcha" [19] and worker platforms such as "Kolotibablo" [20] that have both been covered in the 2010 study have managed to survive the test of time and adapted to the advent of behavioral CAPTCHAs. Interestingly, even the economies of these farms have largely stayed the same. For example, [6] reported a cost of US $2 to effectively solve 1000 text CAPTCHAs. As of the time of writing this manuscript, all CAPTCHA farms we came across in our study were demanding about US $1 - $4 for solving 1000 behavioral CAPTCHAs [19], [21], [22]. We refer interested readers to [6] for more details on the economic analysis of CAPTCHA-farms as it is outside the scope of our work.

3.1. CAPTCHA-farm Operational Mechanics

While the economies of the CAPTCHA-farms have remained the same, their underlying technical operations have been forced to undergo significant changes due to the introduction of behavioral CAPTCHAs. We now systematize knowledge of the working of these farms by studying the APIs offered by 6 different solver API platforms of CAPTCHA farms [19], [21]–[25]. We compiled this list based on multiple internet search queries we performed putting ourselves in the shoes of a potential malicious actor who intends to break CAPTCHA protections on a target website.

A pictorial summary of our findings is in Fig. 2. As before, we use the hypothetical concert ticket sales site,

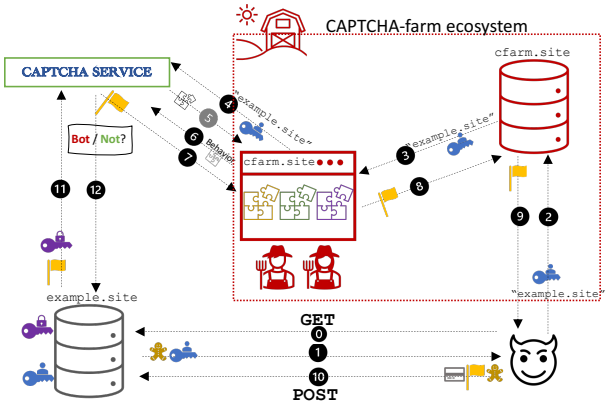

Figure 2: Typical working protocol of a human-driven farm for behavioral CAPTCHAs.

`example.site`. We also assume that the CAPTCHA-farm that the attacker is employing owns and maintains a domain named `cfarm.site` both for their solver API and worker platform services. In the figure, the attacker is shown to be making an API request to `cfarm.site` with metadata about the target site including **site-key** and the **target URL** ❷. These are *essential components of all API requests* that attackers send to CAPTCHA-farms. The worker interfaces of the farms then utilize this information to *simulate* a visit from the target site. This simulation is typically achieved either with the help of a browser extension that a worker is required to install or an instrumented browser. The API documentation of multiple farms explicitly states this simulation behavior [19], [21]. Further, we also confirmed this behavior empirically by building a toy web page with CAPTCHA puzzles and offered them to be solved by various CAPTCHA-farm APIs. Notably, we failed to record a visit from any of the farms to our pages although we received verifiably valid tokens back from the APIs. And finally, we also performed Man-in-the-middle (MITM) network analysis on various worker interfaces to confirm that the `Referer` header of the requests being made to CAPTCHA services is being fabricated to simulate requests from the target website (in ❹) even though the workers' systems never actually visit the target site. Note that such a simulation feature makes good "business sense" for the farms as it enables them to provide CAPTCHA-solving services on web pages that might be behind authentication walls as well.

Once the requests get made to the CAPTCHA service from the workers, the rest of the protocol flow is similar to the legitimate flow as far as the services are concerned. Upon successful receipt of a token the workers' systems simply forward the token to the attacker via the farm's server ❾. The attacker would then typically make a `POST` request submitting the necessary form data (such as credit card and concert event details), HTTP cookie that was set previously, and most importantly, the CAPTCHA token ❿ which will later be verified by the service ⓬.

While the essential nature of the targeted site informa-

tion from the attacker (as opposed to simply a cropped image of the CAPTCHA puzzle as seen in [6]) is a significant change in the mechanics and a cornerstone of our work in this paper, there are some other salient evolutionary changes in the farm ecosystem which are discussed below.

Network Proxy support. First, we noticed that all the farms [19], [21]–[25] now allow attackers to supply network proxy credentials to them. This allows the CAPTCHA-farms to have the same source IP addresses in their requests to the CAPTCHA services (❹, ❻) as the attackers would have when interacting with the targeted site (❶, ❿) and thus becoming less conspicuous to the defenders.

Bi-directional browser fingerprint sharing. We noticed that all the farms allow their customers to supply browser fingerprints that workers should use in their requests (e.g., `User-Agent` header). Further, in cases where the targeted site and the CAPTCHA service belong to the same organization (say, `google.com`), the farm's customers (i.e. attackers) can share the HTTP cookies that have been set by the site to suppress any discrepancies that can be seen [26].

Interestingly, we have also noticed that some farms have provided APIs for sharing the browser fingerprints of their workers to their customers [27]. This is likely to neutralize the support provided by some CAPTCHA services to share the browser fingerprints of the entity that solved the CAPTCHA to their client sites [28]. The key idea here is to enable the bots of the farm customers to "continue" the session by "replaying" the same fingerprints as that of the human worker that solved the puzzle (similar to [29]).

Other new features. As a result of the evolutionary changes in CAPTCHA design (from text to behavioral), the farms have underwent some other changes such as providing support for concurrent requests for the same site, encouraging workers to "enrich" their account histories (such as for `google.com` - to improve scores [30]) and hosting affiliate-based CAPTCHA-breaking software app stores [31].

3.2. Implications of CAPTCHA Protocol Study

Our study of the protocols of behavioral CAPTCHAs and the CAPTCHA-farms as well as the delineation of recent evolutionary changes in these farms has multiple direct implications for web security research as discussed below.

3.2.1. Unrealistic assumptions of prior work. A prior work [32] proposed new designs of CAPTCHAs in an attempt to make them resilient to CAPTCHA farms. Specifically, the work proposed dynamic "gamified" CAPTCHAs which are shown to be resilient to "live relay" attacks (i.e. a frame-by-frame relay of the puzzle from the client to the CAPTCHA-farm workers). However, our study above clearly shows that such relays are not part of the actual operational mechanics of CAPTCHA-farms. Instead, the farm workers directly receive the puzzles from the services and simply return the answer token to the requesting attackers. This means that such gamified designs of CAPTCHAs are also prone to attacks from farms as evidenced by our measurement results in later sections. We thus call upon security researchers to keep these real-world operational mechanics of the CAPTCHAs and their farms in mind when proposing new defensive measures in the future.

3.2.2. Difficulty in combating farms. Our study also shows that CAPTCHA-farms have evolved into a difficult problem to defend against. This is mainly because both the attacker (typically, an automated bot) and the workers (live humans) can intimately collude with each other by sharing network and browser fingerprints. In addition, if the attackers were to route their traffic through a residential VPN network [33], this becomes even more challenging. We state that the endeavor to defend against CAPTCHA-farms is outside the scope of the present work and leave it to future work.

3.2.3. A measurement opportunity. On the other hand, our study does reveal that the operational mechanics of the farm present valuable measurement opportunities which were non-existent in prior CAPTCHA protocols. Specifically, we saw that important site metadata such as the identifying site-key and the targeted site name needs to be *unavoidably* transported to the CAPTCHA-farm worker (❸ in Fig. 2) as part of their working protocol. This means, that if we were to devise a system to automatically simulate CAPTCHA-farm workers, we can potentially *capture valuable real-time measurement data about the websites whose* CAPTCHA *implementations are being attacked*. It is this key intuition that we rely on for the rest of our paper.

4. C-FRAME System

As mentioned in the last section, we wanted to make use of this vantage point that the protocol evolution of CAPTCHA-farms provides to conduct a measurement study characterizing in-the-wild attacks on CAPTCHA deployments which has hitherto never been done before. For this, we designed, implemented and utilized a measurement system that we named "C-FRAME" (CAPTCHA-Farm Reconaissance and Abuse Measurement System). Before implementing C-FRAME, we first laid out three main goals in order to guide our design decisions as enumerated below.

1) First, we desired our measurement system to be **CAPTCHA-service agnostic** in order to not have it be restricted to a specific set of CAPTCHA services.
2) Second, we wanted to perform **large-scale measurements** with our system in order to accurately characterize the nature of CAPTCHA attacks happening globally across a period of time.
3) Last, but most important, we strived to keep our design **ethically considerate** as any measurement system that relies upon dubious ecosystems such as CAPTCHA-farms needs to contend with significant ethical risks [6].

4.1. System design

The main intuition was to design a system that can *simulate* CAPTCHA-farm workers in order to milk in-the-wild

intelligence about CAPTCHA attacks happening in real-time across multiple farms. Therefore, we planned to implement a system to automate the working of multiple CAPTCHA-farm worker interfaces. Here, we faced a key design question. Should we implement a system that can *transparently* emulate a worker (i.e., transparent to the CAPTCHA-farm)? Transparent emulation can be achieved by implementing a system that would take each CAPTCHA problem as a worker and simply forward it to another CAPTCHA-farm as a paying customer. This was the design choice followed by [6]. Unfortunately, this choice is fraught with two ethical risks. First, forwarding CAPTCHA-problems would mean that we would be actively participating in the furtherance of a CAPTCHA attack. Second, as a paying customer, we would be actively injecting money into a dubious ecosystem. While one might argue that the larger research benefits outweigh these risks, it would still limit the scale at which can operate such a transparent system. As scale is one of our main goals, we refrained from using such transparent design approach.

Key design idea. Instead, we devised a simple alternative approach in which we intercept the network requests using a Man-in-the-Middle (MITM) proxy and induce a deliberate change in the site-key. Our approach is illustrated in Fig. 3 where the original site-key (in blue color) is "tampered" (in brown color) before being sent to the CAPTCHA-service for fetching the CAPTCHA-puzzle. As the site-key has been tampered, the service will be unable to confirm that the key matches the site name and hence responds with an error. Because of this, the CAPTCHA-farm worker software will likely assume that there is an issue with the parameters given by their customer. So, it would skip this CAPTCHA-problem and move onto the next problem to be given to the worker.

We note here that we were cognizant of the technical risks that such a design has. If the CAPTCHA-farm has employed an active anomaly detection tool, it might potentially look at the high error rate on our systems and "ban" our simulated worker accounts from their farms. This will lead us into an "arms race" with the farms to either create new accounts or worse, completely halt our data collection[3]. However, we still pursued this risky approach due to its ethical and scale benefits as discussed before.

CAPTCHA-service agnostic design. Having decided on using an approach to manipulate network traffic, our next choice would be to decide where to manipulate the site-key. We had two clear choices for this: either before the site-key reaches the worker software or, on the way from the worker software to the CAPTCHA-service. Manipulating the site-key before it reaches the software would require us to "reverse engineer" the protocol being followed by the CAPTCHA-farm servers to communicate with their workers. However, the advantage with this approach is that we only need to do this step one time for each CAPTCHA-farm. This is because

3. As a preparatory measure to avoid potential measurement data gaps, we also pre-filed a successful IRB application at our university to spend up to US $1000 on CAPTCHA-farm APIs using a transparent version of C-FRAME. Fortunately, we did not need to actually use this approach as none of our were farm worker accounts were ever banned.

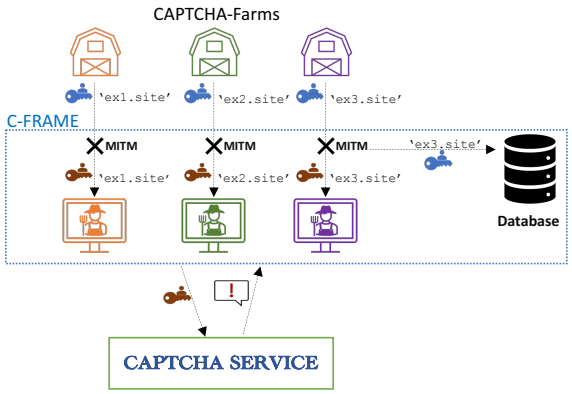

Figure 3: C-FRAME System Overview

it is highly likely that the metadata for all CAPTCHAs will be transported in a similar manner given the high uniformity we observed in both the behavioral CAPTCHA protocols as well as the farm APIs. This means that we can keep our implementation CAPTCHA-service agnostic which was one our main design goals. For this reason, we pursued this approach as indicated in Fig. 3.

4.2. Implementation and setup

To implement C-FRAME, we selected worker platforms for two of the CAPTCHA-farms that whose solver APIs we have studied in the previous section: 2captcha [21] and 9kw [24]. Both of these farms provide Windows-based desktop software for workers which can used after creating accounts with them. We downloaded their software and inspected their network traffic using an MITM tool [34]. Specifically, we inspected the packets coming from the two farm servers to the worker software as discussed previously. We found both the farms to be using HTTPS protocol for communication. The packets were fairly well structured with each request from the farms carrying text fields for: (1) name of the CAPTCHA-service and version number (2) full URL for the web page being targeted (3) associated site-key (4) a unique internal identifier for the CAPTCHA problem being served to the worker. Interestingly, we found both the services to have similar fields. We also performed some ad-hoc testing to confirm that these <site-key, URL> pairs we were collecting in deed reflected real site data. Based on this background knowledge, we were thus able to implement C-FRAME for both the farms we considered.

We built C-FRAME to run inside multiple VMs each running an instance of the worker software. The data from multiple VMs would be offloaded to a database in real-time. In particular, we stored the aforementioned four data points as well as timestamp information in a centralized database. We simulated periodic user inputs (mouse movements) in these VMs as our pilot studies showed that otherwise, both the worker tools were "pausing" the software thus hindering our data collection attempts.

We used Python to implement the core parts of C-FRAME with MongoDB as our database server. We used 4 Xen-based Windows 10 Pro VMs, each with 4 GB of RAM. We utilized 3 of these for running 3 instances of 2captcha software while 1 was used for running 9kw's worker software. We based this decision based on a pilot run in which we saw more CAPTCHA-solving tasks from the 2captcha ecosystem. For this, we created 4 different worker accounts (3 with 2captcha, 1 with 9kw) using throw-away email accounts. We intend to make all our source code available to security researchers upon request.

4.3. Overview of collected attack data

Timeline. Our data collection phase commenced on March 6th, 2023, and ended on August 6th, 2023. Unfortunately, a major security incident caused an outage of our entire university's network for a period of two months in between. This affected C-FRAME's ability to collect attack data. As a result, in the end, we were only able to collect CAPTCHA attack data over a period of 92 days using C-FRAME.

Note that none of the four worker accounts have ever been suspended by any of the farms during this entire period. Furthermore, all four VMs have continued to independently collect a significant amount of data over this time period.

Collected CAPTCHA attack summary. During this 92-day time period, C-FRAME collected a large dataset of global CAPTCHA attack incidents comprising 425,257 unique CAPTCHA-solving requests and 42,612 unique URLs. Notice that many of the same URLs are being attacked multiple times as might be expected. These requests are associated with 1,417 fully qualified domain names (FQDNs) and 1285 eTLD+1 domains[4]. The most popular eTLD+1 domain in D_{All} was `twitter.com` attracting as many as 222,679 requests to solve CAPTCHAs on their site.

For deeper analysis, we also found it useful to merge FQDNs into "organizations" by ignoring the eTLD similar to prior work [35]. In other words, we consider the effective second-level domain name as an organization irrespective of differences in the effective top-level domain name. We term these as "org"s in this paper to avoid confusion with eTLD+1. For example, `airbnb.fr` and `airbnb.co.cr` both got merged into a single org named `airbnb` as a result of this merging process. In our collected dataset, the 1,417 FQDNs merged into a total of 1023 orgs.

We denote this entire dataset that we collected as D_{All} in contrast to a smaller dataset D_{QA} that we will describe later in the paper. A summary of the above numbers are also reported in Table 1 for convenience. This complex and rich attack dataset necessitated both qualitative and quantitative analysis in order to characterize and measure that attack data which we describe in the following sections.

4. The eTLD+1 domain for `www.bbc.co.uk` is `bbc.co.uk`. We rely on Mozilla's Public Suffix List to make this determination

5. Characterizing CAPTCHA attacks

Research questions. Given the enormity of the attack data we collected using C-FRAME, we wanted to develop a systematic approach to first understand the context behind these attacks. It should be noted that, to our knowledge, there exists no prior measurement work that analyses the types of web pages on the internet that are seeking protections from CAPTCHA services. The only systematic CAPTCHA measurement work that we are aware of thus far has only measured the prevalence of CAPTCHA protections on "account-creation" pages on top sites [4]. Therefore, several questions remain unanswered not only about CAPTCHA-attacks but also about the nature of CAPTCHA deployments in general. For example, what other parts of web pages on the internet host CAPTCHA protections? What are the intents and objectives of the CAPTCHA attackers? What are the most popular targets of such CAPTCHA attacks? Can we divide these attacks into various categories based on the targets? How is the geographical and temporal distribution of these attacks? These are some of the research questions for which we sought answers by using the data we collected.

We wanted to utilize our dataset to seek answers to these questions. As explained in Section 4.2, we had at our disposal the full URL associated with all the attack requests. As a pilot experiment, we attempted to automatically analyze the content hosted on these URLs as a first step to seeking the above answers. However, we quickly realized that this is an inadequate approach as we ran into a number of issues. For example, we noticed that a large number of URLs are *single-use URLs* which typically cannot be re-rendered. Sometimes, this could also be because the web application might need the original session cookie in order to re-render the page. While in most cases, an analyst might be able to gain context behind the CAPTCHA protection and attack by revisiting a parent path, it is beyond the capabilities of a heuristic and automated data analysis approach.

Key idea. Recognizing the limitations of automated analysis approaches, we look for alternative approaches. Grounded-theory based qualitative analysis (QA) approaches have been successfully used in many prior security research works [36], [37]. In these cases, researchers popularly rely on semi-structured interview data to extract codes and characterize various security phenomenon relating to user behavior. While we do not have any interview data in our research, the overarching goal of attempting to extract themes and theories from the data still remains the same. Noticing this, we implemented a qualitative analysis approach using Grounded Theory [38] for our study as well. We describe this approach in detail below.

5.1. Analyst data retrieval

Grounded Theory based QA approaches typically involve the efforts of multiple independent analysts who work together on semi-structured datasets to extract themes/codes from them. Unfortunately, in our case, the data that we collected directly from C-FRAME only gives us the full URLs

for the CAPTCHA attacks and their associated timestamps. This, by itself, is not sufficient data to provide enough context for qualitative analysis. Therefore, based on the findings from some pilot analysis, we proposed a series of steps for each analyst to follow in order to gain more context for each of the attack URLs seen in our dataset. Each analyst during our manual analysis will follow these steps to draft and record "contextual notes" about the URLs that they are analyzing. These notes will then provide the semi-structured data that can utilized in a traditional QA approach. Below, we enumerate these steps with the help of some real illustrative examples come across in our study. One clarifying note before we begin enumeration is that the all analysts in our study have had easy access to a cluster of all related URLs and timestamps. Their task was to categorize these clusters of URLs (as opposed to a single URL). We will describe this clustering process and the rationale behind it in more detail later.

The steps taken by analysts can be divided into three phases as below.

Phase-1: URL access attempts. The goal of this phase for the analyst is to make various attempts to access the URLs that make up the current cluster being examined.

1) The analyst will first attempt to visit some URLs in a given cluster directly. If successful, they will make notes about the web page that they visited. For example, if the URLs are part of a large cluster that lead to a web page hosting a pirated movie, then it becomes obvious to the analyst that this is part of an attacker's attempt to "steal" pirated content.
2) If URLs are inaccessible, the analyst will attempt to use the approach of "backtracking" the URL path. Specifically, the analyst will attempt to strip the URL's path from the right side and make iterative visit attempts. When doing so, they will also consider the URL path elements being stripped off, to make sure no vital context is being lost (thereby mistakenly leading to a page with a different context).
3) Another alternative approach the analyst will attempt for accessing URLs is to supply the full URL as a query string to a search engine to see if this can lead them into visiting a highly similar URL.
4) Based on their discretion, the analyst will also attempt to use a VPN to visit the URL from a specific country. In our pilot experience, we noticed that some attacked URLs were only accessible via VPN from specific countries (such as, Brazil). For any sites that use a foreign language that the analyst does not speak, the analyst will use a browser-based language translation tool. When doing this, the analyst will make note of the language.

At the end of the above steps, the analyst will make separate notes about their success or failure in visiting the URLs in question and the procedure they used for it.

Phase-2: Geographical tagging. If successful in being able to access the URLs (or similar URLs), the analyst will also look for any association with a particular country and make note of it if that exists. While this might have some variation based on the discretion of the analysts, we argue that this will yield a much higher fidelity data point than heuristic reliance on a data point such as ccTLD. `www.salonboard.com` (Japan) and `kktix.com` (Hong Kong) are a couple of attack target examples in our data which would have resulted in false negative with a ccTLD approach. On the other hand, `mixdrop.sh` is an example of a file-sharing site that would have resulted in a false negative. The analyst will also make a note of any sub-country level geographical affinity (such as state). In particular, we have seen specific government sites of the US and Brazil be subjected to CAPTCHA attacks as we discuss later.

Phase-3: Third-party context gathering. The goal of this phase is to make various attempts to gain more context about the attacks being performed. This phase can be pursued regardless of the outcome of the previous steps.

1) The analysts will attempt to search in the CAPTCHA-farm app stores [31] for the existence of any app that purports support for the current domain name in question. Often, these apps will list the features (such as the ability to create mass social-media accounts, or buy "limited release" goods) that also provide vital context to the analyst about the attacks in question which will be added to the analyst's notes. The analyst will also make separate note of this as a "potential attack traffic source" for the candidate domain.
2) The analyst will also make queries in source code search engines (such as on `github.com`) which include both the domain name in question and the API code snippets of CAPTCHA-farms in order to get access to any code repositories that might be using them to generate similar attack requests. For example: [39]–[41].
3) The analyst will make search engine queries for the topic of bot attacks on the domain in question to gain more perspective from news articles, social media accounts about the usecases for the attacks on a given domain.

The above described phased data retrieval procedure will provide our analysts with vital textual data and references which will serve as input for the qualitative analysis.

5.2. Qualitative analysis data pipeline

Having described our core semi-structured data retrieval process first, we can now provide end-to-end details of how this process fit into our Qualitative analysis (QA) pipeline.

QA dataset. Three of the paper's authors have served as analysts for this work. Previously, we saw that we had a total of 425,257 unique requests in the D_{All} dataset as described in Table 1. As the elaborate steps that an analyst needs to pursue for data retrieval would be time-intensive, we wanted to limit our manual analysis to only those sites that were being targeted with a high-frequency. For this reason, we decided to create a subset of our collected dataset that we term as D_{QA} for the QA process. To do this, we simply eliminated all orgs (as defined in Section 4.3) with less than 5 requests. This new dataset is also shown in Table 1. Notably, this new dataset retained 98.6% of the requests

(419,648) while reducing the number of orgs to be analyzed to 361 ($\approx$ 65% reduction). Thus, we greatly reduced the analyst load while retaining most of the attack data.

Name	# requests	# URLs	# FQDNs	# eTLD+1s	# orgs
D_{All}	425,257	42,612	1417	1285	1023
D_{QA}	419,648	41,226	602	457	361

TABLE 1: Datasets collected and analyzed in this study

Org-based batches. We followed an org-centric approach when assigning data load to the analysts. Specifically, we divided the 361 orgs in to a set of about 20 equal-sized "batches". These batches were the units (of about data from 18 orgs each) that were given to each analyst for analysis. The org-centric approach ensured that all the data related to an org is put in the same batch and analyzed at the same time. The intuition behind this is that when CAPTCHAs of multiple sites in the same org get attacked (say, `expedia.es` and `expedia.com`), it is likely that their contextual clues as well as the theories that can be derived from them are similar. Hence, it is beneficial to keep these requests together in an effort to optimize analyst time.

Agglomerative clustering. Each batch comprised of text files that contain the URL as well as request timestamps. As can be seen from Table 1, some orgs in such a large dataset will invariably end up having hundreds of URLs. `sony` for example, was the org with the largest number of URLs (5199 URLs). It turned out that a lot of these URLs were referring to attempts by attackers to automatically login to PlayStation accounts. However, since Sony uses the aforementioned *single-use URLs* during their login process, most attack attempts triggered unique URLs thus resulting in a large number of URLs to be analyzed. In order to optimize analysts' time in such cases, we utilized an agglomerative clustering approach [42] based on the query parameters in the URLs. In the case of Sony, for example, this aggregated the 5199 URLs into 8 clusters. This allows the analysts to pick and choose URL samples across different clusters quickly instead of being lost in a large number of URLs all with the same patterns (and attack usecases). Note that this clustering only affected the ordering and grouping of the URLs as an assistive measure. All the raw URLs and the timestamp data were still present in the text files for the analysts to inspect if needed.

Among the 20 batches described earlier, 18 were used for a "collaborative coding" process. The last two batches were retained for an "inter-rater reliability testing" process.

Collaborative coding. This collaborative coding process forms the bulk of the QA work done in this study as it included analysis of about 90% of orgs in the D_{QA} dataset. As mentioned previously, each of the 3 authors served as an analyst. After agreeing on the all the steps for the data retrieval (Section 5.1), each of the 3 analysts were allotted some of the batches. For each batch, the analysts followed all the data retrieval steps and applied open coding methodology on the obtained notes to derive codes representing various categories of CAPTCHA attacks for each org. Each batch was analyzed by at least 2 analysts independently. After every 4 batches, all 3 analysts met up together in order to perform axial coding which included merging/splitting of categories and detailed discussion about conflicts regarding each batch. All the notes from different analysts were merged together during these meetings and compiled into a comprehensive codebook which was maintained as a spreadsheet with all the references and notes. In some cases, despite the best efforts, the analysts were unable to gather much information about the potential motivation behind the attacks. In such cases, the analysts agree to mark the case as an "unknown".

Upon completion of 18 batches of collaborative open coding stage, the analysts performed one last axial coding step to modify, eliminate, split or merge various attack categories and other labels (such as geo-tags), create any meta-categories (to be described later) to best represent similar use cases and refine the codebook. At the end of this stage, the QA analysis process yielded 34 distinct CAPTCHA attack categories characterizing various attacks that we collected. The analysts grouped them into 4 meta-categories showing a varying scale of severity of threat intent: fraud, abuse, spam/scam, and gray/benign. These 34 attack categories are shown in Table 2 and will be described in the next section.

Inter-rater reliability testing. The remaining two batches comprising about 10% of the 361 orgs in the D_{QA} dataset were utilized for inter-rater reliability testing between the 3 analysts. After being given the batches, the analysts relied solely on the codebook and analyzed the remaining two batches in a complete independent fashion. The analysts agreed to not develop any more new codes during this time but instead simply attempt to map each of the CAPTCHA attack instances to existing 34 attack categories or mark a case as 'unknown'. At the end, we computed Krippendorff's α coefficient [43] for the labels assigned by each of the analysts. The α coefficient we obtained was about 0.88 indicating a very high level of agreement between the 3 analysts [44]. Furthermore, a CDF of the time of creation of these 34 attack categories across time (Fig. 4) during the QA process shows that we reach an early level of saturation in 36 days. This shows the stability of these attack categories as they yield well to a large number of ensuing new attack requests in the following days.

Analyst effort. We measured the total time spent by the analysts during the open and axial coding stages as well as during the inter-rater reliability testing period to be in excess of 200 hours.

6. CAPTCHA Attack Categories

We provide more details on the 34 attack categories under the 4 meta-categories by discussing some exemplary cases under each category based on the analysts' notes.

6.1. Fraud Category

6 attack categories can fall under the "Fraud" meta-category of as shown in Table 2. While this is not the most popular attack meta-category, it does represent the most

Use-Case	# Reqs	# FQDNs	Target example
Fraud Category			
Streaming fraud	5272	3	deezer.com,twitch.tv
Event ticket bot	1632	17	tickets.rugbyworldcup.com,ticketmaster.com
Identity fraud	845	11	bankofamerica.com,apps.twc.state.tx.us
Card fraud	307	10	homedepot.com,giftcardmall.com
Settlement lawsuit fraud	21	3	mwpfsettlement.com,refererheadersettlement.com
Survey fraud	7	1	qualtrics.com
Total	8084	45	
Abuse Category			
Pirated content download	24,652	70	turbobit.net,uploadgig.com
Offer abuse	11,345	23	platform.openai.com,samsung.com
Travel site abuse	9668	51	expedia.es,airbnb.com
Cryptocurrency abuse	7197	36	claimbits.io,bingx.com
Amazon flex bot	6739	1	amazon.com
Content theft	2923	10	onlyfans.com,discovery.com
Auto bypass shorteners	2185	20	mexa.sh,cutty.app
Voting bot	581	2	gtop100.com,registraduria.gov.co
Collectible bot	428	4	pokemoncenter-online.com,funkoeurope.com
Fashion apparel bot	312	11	crtz.xyz,nike.com
Marketplace bot	215	5	tiktok.com,poshmark.com
Appointment bot	197	7	driverpracticaltest.dvsa.gov.uk,visa.vfsglobal.com
Flight ticket bot	107	5	wizzair.com,bangkokair.com
Total	66,549	245	
Spam/Scam Category			
Social media automation	276,809	16	twitter.com,linkedin.com
Gaming automation	30,153	25	roblox.com,leagueoflegends.com
Dating bot	10,395	8	ourtime.com,tinder.com
Email automation	9533	8	live.com,rambler.ru
Hosting automation	341	4	github.com,dropbox.com
Potential comment spam attack	11	2	beepworld.de,helpshift.com
Total	327,242	63	
Gray/Benign Category			
Government data download	7064	79	cnd.pbh.gov.br,micourt.courts.michigan.gov
Developer testing	1798	3	mysite.com,geetest.com
Shopping scraping	368	7	electroprecio.com,cdiscount.com
People search	321	4	familytreenow.com,truepeoplesearch.com
Article data scraping	164	3	sevenjournals.com,webhostingpost.com
Search or ads optimization	84	6	google.com,google.fr
Company data download	69	6	immobilienscout24.de,system.reins.jp
Personal data removal request	19	2	idtrue.com,thatsthem.com
Report pirated content	5	1	alliance4creativity.com
Total	9892	111	

TABLE 2: List of Attack Categories / Use Cases

egregious nature of attacks we have seen in our study often leading to enactment of governmental and regulations aimed at curbing these attacks. Some are discussed below.

Event ticket fraud. Ticket bots have long been a pervasive issue, leading to the enactment of the Better Online Ticket Sales (BOTS) Act[45] in the U.S. in 2016, which prohibits the circumvention of security measures and the resale of tickets obtained through such means. Despite legislative efforts, ticket bots continue to persist. During our monitoring period, we observed attacks on 17 event ticket FQDNs across six countries: Brazil, US, UK, Germany, Italy, and France. These attacks targeted highly sought-after events in 2023. For instance, on 06-22-2023, we documented approximately 90 requests for "Tickets for Fun", a prominent ticket seller in Brazil for Taylor Swift's Era tour concerts [46]. Further, the time window for the opening of ticket sales exactly matches the times seen in our logs [47] [48]. Interesting, in response to the chaos caused by these ticket bots during Taylor Swift's tour, a Brazilian lawmaker introduced a bill, named after Taylor Swift, to outlaw ticket bots [49]. Besides Taylor Swift's Era tour, Coldplay's "Music of The Spheres World Tour 2024" was heavily targeted on `ticketmaster` org across Germany, Italy, and Denmark, with over 500 requests during July 2023, again coinciding with the concert sale times [50].

Another notable event in our data was the Rugby World Cup 2023 [51] took place in France. For two months June and July, we observed a sudden peak for `tickets.rugbyworldcup.com` with nearly 900 requests in total for matches. The URL data included the exact matches for which the tickets were being bought. UK football clubs Arsenal and Tottenham were also found to be victims of ticket bots in July.

Streaming fraud. Fraud activities in the streaming industry pose a significant threat to both content creators and audiences. Streaming fraud involves artificially inflating streaming numbers on music tracks or videos using bots [52]. While this practice can generate illicit revenue and manipulate popularity, it is also unjust and illegal. Legitimate artists are adversely affected by this as it undermines their income and distorts their performance statistics [53]. Our analysis has revealed that Deezer, Spotify, and Twitch platforms are prime targets for streaming fraud. We documented 5272 requests for account creation and login URLs across these three platforms. Streaming fraud was a persistent issue for Deezer and Spotify, with the potential to generate over half a billion dollars annually through these platforms [54].

Settlement fraud. Class-action lawsuits are legal actions brought by a group of individuals with similar claims against one or more defendants, such as customers, employees, investors, or patients. When such a lawsuit is successfully resolved, the resulting proceeds are known as a class-action lawsuit settlement [55]. These settlements are typically detailed on a dedicated website, outlining the specifics of the lawsuit, eligibility criteria for affected individuals, the submission deadline, and the process for making a claim. To prevent automated fraudulent claim submissions at scale, CAPTCHAs are commonly employed. Our analysis identified three settlement lawsuits being targeted for attacks. A notable example was the website setup to handle the "Referer header settlement" lawsuit against Google involving US $23 million for improper sharing of search queries with third-party websites [56]. We also found fraudulent claim attempts for taking part in class action against a pet foods [57] and a

fashion store in the US [58]. Notably, in all these cases the requests occurred right before the claim deadlines listed on those sites.

Other categories. Further, we uncovered three distinct but equally concerning trends. First, *survey fraud*, particularly on platforms like Qualtrics, where bots skew survey results, leading to researchers questioning the viability of online surveys. For instance, studies from Stony Brook University and the University of Michigan were seen being attacked undermining their survey quality. Second, *identity fraud* emerged as a significant issue, especially in the form of credential stuffing attacks. This was evident in attempts to exploit unemployment benefit systems in Texas and Georgia and in Brazil's digital service platforms, where CPF (a personal identifier) fraud is rampant. Such activities not only compromise individual data but also strain public resources. Last, *card fraud*, manifested through unauthorized checking and reloading of gift card balances, was prevalent across various retailers and financial institutions. This included attacks on sites such as `homedepot.com` and `razer.com`.

6.2. Abuse Category

"Abuse" is our widest meta-category with 13 categories in them often refering to apparent attempts by the attacker to gain an unfair benefit or advantage over other actors.

Amazon Flex Bots. Amazon Flex operates similar to many gig economy platforms, where workers log in and manually select available jobs in real-time. Amazon Flex drivers have been known to be using bots to cheat their way to get more work [59], [60]. We received 6739 requests for Amazon Flex jobs during our collection time.

Appointment Bots. Appointment bots serve the purpose of swiftly securing difficult-to-obtain appointments, often reselling these coveted slots for a fee. Visa appointments, known for their limited availability, contribute to prolonged waiting times, prompting the creation and use of appointment bots either for personal use or resale. Our data identified four websites facilitating Spain visa applications in four different countries (Algeria, the UK, Russia, and China) being targeted for attacks. VFS Global, a prominent visa outsourcing and technology services specialist, was also a target. Additionally, we captured attacks on a web page for a restricted New Zealand working holiday visa, featuring a mere 2-hour application window [61].

A parallel scenario involves driving test appointments in the UK, booked through `driverpracticaltest.dvsa.gov.uk`. Excessive waiting times have led to a surge in the use of automated bots for such appointments [62], [63].

Cryptocurrency Abuse. Cryptocurrency abuse and offer abuse encompass the deployment of bots to exploit features and vulnerabilities within cryptocurrency platforms for illicit financial gains. Cryptocurrency bots, for instance, engage in practices such as pump-and-dump schemes. Our dataset includes data from seven distinct crypto-exchange platforms, including TFX, Kucoin, and Coinlist. Specifically, URLs related to Kucoin pointed to "Burning Drop" events, a feature designed to benefit users holding KuCoin Shares (KCS), where KCS can be burned in exchange for certain cryptocurrencies [64]. Notably, we observed 69 requests on 11/05/2023, aligning with the date of announced events. TFX, represented by Triumphfx's cryptocurrency coin, faced a loss in capital value, leading to customer profit losses. The demand for coin settlement, a process enabling customers to sell/buy the coin at a specific rate, resulted in roughly 20 requests to log in and perform coin settlement [65].

Beyond crypto abuse, various other forms of abuse we encountered include faucet giveaways, raffle lotteries, and web3 token exploitation. Bots are employed to automatically collect rewards by completing specified tasks, such as sharing events on social media or participating in games. Our dataset identified over 25 FQDNs associated with this form of crypto abuse, amassing 5500 requests in total.

Offer Abuse. Offer abuse involves bots exploiting opportunities, often seen in trial periods or coupon codes offered by companies. This form of abuse frequently involves mass account creation to capitalize on promotional offers. Notably, companies like EA Sports, offering limited beta game versions on a first-come, first-served approach were targeted.

6.3. Spam/Scam Category

This meta-category represents instances where spammers exploit legitimate services as channels to distribute their content, thereby gaining user trust for subsequent scams. Often, the attackers create a large number of accounts on their target sites which are widely popular in order to further their spam/scam actions. Thus, this is the most popular meta-category in our attack dataset.

Our analysis identified Twitter as the primary target for such account creation attempts, with the highest number of 255,480 requests per domain seen in our dataset. Twitter accounts created en masse via these CAPTCHA attacks are likely to be used in scams as seen in a recent work [66]. Other affected social media platforms include LinkedIn, Snapchat, Discord, and Kik along with email providers like Outlook and GMX. Dating and hosting platforms, such as Tinder and Github also saw substantial bot account creation activity. Further, our data also revealed botting in gaming industry, including Roblox, Sony-PlayStation, EA Sports, Battle, and League of Legends. These bots are created en masse and designed to level up player accounts before being sold for profit. We observed 30,153 requests recorded across 25 gaming FQDNs.

6.4. Gray/Benign Category

Interestingly, we also noticed several instances where CAPTCHA-farm services were used for data scraping, not necessarily with confirmed malicious intent, but rather to enhance business operations or for personal purposes. While data scraping can be a nuisance to businesses, it is generally legal and typically doesn't disrupt services as severely as other categories might.

Our analysis identified a total of 9 categories related to data scraping activities. A significant focus was on government websites from 11 countries, including Brazil and the US, where bots scraped judicial, tax, and licensing data from 79 different sites, resulting in 7064 requests. E-commerce platforms like France-based Cdiscount also faced scraping of product information, totaling 368 requests. Personal data scraping was evident on platforms like `truepeoplesearch.com`, with 321 requests for information like names and addresses. Additionally, bots targeted news sites for blog and article content, as well as company data from businesses like Rclgroup in shipping. Real estate platforms like `immobilienscout24.de` and `system.reins.jp` were also scraped for data. These instances highlight the extent of non-malicious data scraping, ranging from government records to commercial and personal information.

7. Measurement Results

This section provides some quantitative analysis of the attack data that we collected in the study.

CAPTCHA farm	# requests	# eTLD+1s
2captcha	380,617	1125
9kw	44,640	173
Total	425,257	1285

TABLE 3: Breakdown of CAPTCHA attacks by farm

7.1. CAPTCHA-Based Analysis

Table 3 shows the breakdown of attack requests and targeted eTLD+1s by the source CAPTCHA-farm. As can be seen, 2captcha was responsible for a majority of the requests but 9kw also contributed towards a significant number of attack requests. We can also see the time-based distribution in Fig. 4. Table 4 shows statistics related to number of distinct eTLD+1s seen across different number of worker accounts in 2captcha. Each row indicates the statistics for the number of attack requests from the PoV of 1/2/3 workers. We can observe that there is minimal loss in the number of collected eTLDs with even less than 3 workers. Although the minimum and maximum numbers of eTLD+1s seen do increase with more workers, there is no significant increase in information when going from 2 to 3 workers. This hints that beyond our configuration of 3 workers, adding more workers might not likely provide a significant increase in the number of eTLDs (i.e. attack data) that can be captured.

# workers	Min # eTLD+1s	Max # eTLD+1s	Median # eTLD+1s
1	553	839	600
2	789	1002	995
3	1125	1125	1125

TABLE 4: # eTLD+1s seen: PoV of various # workers

Additionally, we obtained requests corresponding to more than 12 CAPTCHA services as shown in Table 5 illustrating the CAPTCHA-agnostic nature of C-FRAME.

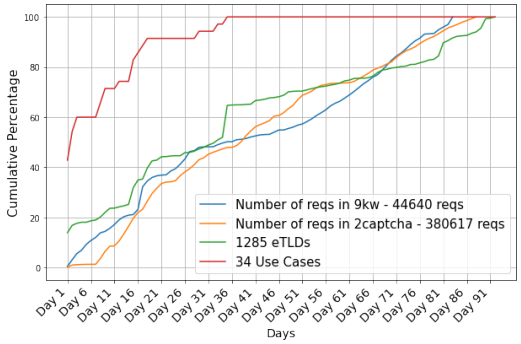

Figure 4: CDF for # of requests for each farm, total categories, and eTLD+1s

Funcaptcha, owned by Arkose Labs, emerged as the most targeted type in our data. This prevalence is attributed to Twitter, the most targeted platform, utilizing Funcaptcha as the front-tier defense against bots. Funcaptcha boasts a substantial presence in the top 1000 domains, including Snapchat, Roblox, and LinkedIn among others. Google's Recaptcha (v2 + v3) on the other hand, accounts for the highest number of affected eTLD+1s amongst all services.

CAPTCHA service	# requests	# eTLDs
funcaptcha	355,111	156
recaptcha	63,524	846
hcaptcha	4507	210
amazon_waf	908	6
geetest	433	32
yandex	252	4
turnstile	205	28
geetest_v4	193	10
tiktok	55	1
capy	49	4
lemin	8	3
Others	7	3
keycaptcha	5	5
Total	425,257	1285

TABLE 5: Breakdown of CAPTCHA attacks by service

7.2. Time-Based Analysis

We next focus on identifying patterns that occur across multiple points of time within our collected dataset. Notably, we observed a steady volume of requests for the two CAPTCHA farms over several days, as illustrated in Fig. 4. We plotted a CDF of the distinct 1285 eTLD+1s encountered over time as shown in Fig. 4. Barring a few steep parts of this plot we observe steady increments in percentage of domains indicating regular introduction of new domains throughout our data collection period. Also, a considerable proportion of domains, specifically, over 23%, were subject to repeated requests across more than 10 days. Table 6 (in Appendix) lists the top 10 domains across 34 categories and 4 meta-categories that were repeatedly targeted for bypassing CAPTCHAs.

Similarly, we aimed to conduct a temporal analysis on the 34 attack categories identified through the Qualitative Analysis (QA) pipeline, as outlined in the previous section.

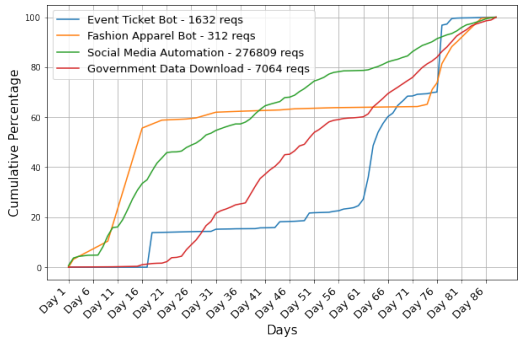

Figure 5: CDF for requests in some attack categories

Fig. 4 illustrates the CDF of the distinct attack categories over the data collection period. Unlike the domains, the increase in the number of identified categories levels off after 34 days. This finding suggests that although there is a regular increase in the number of targeted domains, the majority are associated with one of the already identified attack categories. This shows the stability of the attack categories that we identified and lays a strong case for future exploration into each of the CAPTCHA attack categories.

Fig. 5 illustrates the temporal dynamics of four sample attack categories. We observed distinctive patterns in the Fashion Apparel Bot attack category, revealing two distinct phases. Upon closer examination, these phases align with two separate events. The first attack occurred in mid-March 2023 and targeted `starcowparis.com`, `eu.kith.com`, and `cruz.xyz`, resulting in 124 requests. Subsequently, a second attack unfolded from mid-July to early August 2023, targeting `nike.com`, `offspring.co.uk`, `shopnikcekicks.com`, and `nike.com.hk`, with a total of 54 requests. It is noteworthy that these attacks coincided exactly with the respective platforms' collection release dates and times during these periods.

Another discernible step pattern emerges in the Event Ticket Bots category. The first step on day-17 comprised over 216 requests, targeting the sale of tickets for a TV show celebrity meetup event on the event ticket platform, Kktix in Hong Kong in May 2023. The second step seen between the days of 61 and 71 was for purchasing the tickets of two Taylor Swift concerts in Brazil that were up for sale in June 2023. The third steep step in the graph refers to 419 attempts to break CAPTCHAs on a `ticketmaster` org page (with different TLDs such as `.eu`, `.de`, `.fr` and `.dk`) on day-76 for the sale of Coldplay concert tickets in Europe.

7.3. Location-Based Analysis

Here, we present the geo-location based analysis of our collected dataset. First, we consider the geolocation mapping that was manually done by the analysts during the QA process, wherein they record the country and state of the targeted domain if it can be inferred for the org in question. An analysis of these tags quickly reveals that the attacks have a worldwide reach, affecting 58 countries across 5 continents with web sites in 15 different languages. Brazil and the US were identified as particularly prevalent areas, with numerous requests. Specifically, in Brazil, the domains spanned over 11 states, notably Rio De Janeiro and Sao Paulo. In the United States, over 20 states were involved, with a marked emphasis on government data scraping. This illustrates the global nature of present-day bot attacks being perpetrated via CAPTCHA-farms.

Next, we also re-implemented the approach outlined in a prior work [35] using the Google CRUX dataset. This approach and the dataset provides a way to compute an "endemicity" score for each domain name based on its relative popularity across multiple countries in order to determine if it is a global or a national domain. Our implementation of this approach revealed that only 81eTLD+1s (out of 1285) in our dataset were globally popular, thus showing that a large number of CAPTCHA attack requests were targeting regional/nationally popular domains.

8. Limitations

Although C-FRAME was effective in collecting real world CAPTCHA attack data, we acknowledge the vulnerability of its current design to potential evasive measures by CAPTCHA-farm operators in the future. For example, the farms can deploy anomaly detection measures to automatically identify C-FRAME's worker accounts and ban them. In such cases, as discussed earlier, the design of C-FRAME can fall back to a fully transparent mode which relays CAPTCHA challenges to a parallel solver API platform while simultaneously collecting attack data. However, the usage of such a design should be restricted due to ethical considerations. Another approach is to pursue collaboration with CAPTCHA services, which could offer more options, such as returning a trackable token as noted in Appendix B.

While our design does not let any particular CAPTCHA problem to be solved, the farm's server might likely relay the challenge to another real worker who might ultimately solve the challenge. At the same time, this does not necessarily mean that the site operators will not be able to identify the attack. We admit that it is in deed possible for site operators to have identified some orthogonal suspicious signals (based on post or pre-CAPTCHA solving actions of the attackers on the site) to help them in identifying the attack requests despite the CAPTCHA services' insistence that this is a genuine request. Thus, the precise success rates of our observed CAPTCHA attack attempts remain uncertain. However, the advent of anti-bot legal frameworks, extended media coverage on many of the CAPTCHA attack categories we cited, responses to our disclosure (see Appendix A), all affirm the gravity that farm-driven CAPTCHA attacks pose to the industry in general. We also contend that such a farm economy cannot thrive unless there is a profit motive for malicious actors where they ultimately earn more than what the pay for the usage of solver APIs.

9. Related Work

Prior work on CAPTCHAs. Most prior CAPTCHA-related research works have focused on static text CAPTCHAs, a previously popular version of CAPTCHAs. While some of these studies have cited usability issues [2], [3] associated with text CAPTCHAs, others have shed light on their susceptibility to ML-based attacks [7]–[15]. As a result, the industry has gravitated towards deploying modern behavioral CAPTCHAs which allows for capturing the behavior of the user in addition to the solution of the CAPTCHA puzzle (see Section 2). To our knowledge, no prior study has yet systematized the operational mechanics of these modern CAPTCHA variants. More recently, a study has cited usability issues associated with these modern CAPTCHAs [4] but did not delve into their operational mechanics.

Prior work on CAPTCHA-farms ([6]). C-FRAME leverages human-powered CAPTCHA-farms to gather data about in-the-wild behavioral CAPTCHA attacks. In 2010, Motoyama et al. [6] focused primarily on studying the efficiency (solve rates and speeds) and worker characteristics (labor cost, country of origin etc.) of these CAPTCHA farms. Note that this was published prior to the introduction of modern behavioral CAPTCHAs in 2013. In order to gather these findings, Motoyama et al. relay static text CAPTCHA problems collected as a "worker" to another farm's solver APIs. In our study, we leverage the behavioral CAPTCHA protocols during our worker emulation to simulate a "fake error" in the worker software and thereby avoid the ethical ramifications associated with a relay-based design. More importantly, the modern evolution of CAPTCHAs allowed to us to collect, for the first time, fine-grained data about the target attack URLs which was not possible in [6]. Nevertheless, as a minor contribution, Motoyama et al. [6] have indeed attempted to manually infer the targeted sites by conducting visual inspection of CAPTCHA problems received as a worker. This approach has the following limitations:

1) Visual inspection method only enables the identification of the CAPTCHA service provider and not the targeted site. If multiple websites carry CAPTCHAs of the same style, it would be impossible to identify the targeted site. Furthermore, if two CAPTCHA-services were to utilize similar designs, it would also be impossible to differentiate between them.
2) By design, such visual inspection will not help in identification of CAPTCHAs with styles that are not preknown to the analysts.
3) Such a visual inference process is also prone to errors by analysts and is not scalable as it involves manual effort. In [6], only about 20 attacked sites have been inferred on each farm.
4) Finally, visual inspection of image CAPTCHA problems will not yield data about the targeted URLs thereby missing important context about the attacks.

In contrast to the above, in our work, we study and rely on the operational protocols of modern behavioral CAPTCHAs and their farms to reliably, scalably and ethically collect a large-scale dataset of real-world site URLs (42,612 unique URLs spanning 1417 sites) that have been targeted by CAPTCHA attacks.

Prior work on bot attacks. Our 90-day deployment of C-FRAME allowed us to collect data about CAPTCHA attacks on targeted sites in the real-world. As CAPTCHAs are typically deployed to prevent large-scale bot visits, this dataset gave us a window to observe real-world bot attacks. In this context, our work complements other web-based bot attack research pursued in the past. Notably, a large amount of prior work focused on infiltrating botnets (either as a bot or a bot herder) to measure their activities [67]–[73]. In this work, we complement this by choosing CAPTCHA-farms as a vantage point instead of botnets for observing web bot attacks. Our findings show that we are able to discover several new categories of attacks (such as ticket scalping attempts) which have not been seen in the attack categories of prior botnet-related works [70], [71], [73]. Other works have focused on collecting bot data from the point-of-view of singular organizations [74]–[80]. In contrast, our C-FRAME system allows us to collect web bot attack target data across multiple organizations. Another line of anti-bot research has focused on collecting data from honeypot-based approaches [81]–[83]. Although these approaches are good for studying general-purpose web bot attacks, attackers might not have the same financial incentives to target honeypot-sites as they do for a real-world site. Our findings thus complement these results by discussing real-world web bot attack data.

10. Conclusion

In this paper, we design and implement C-FRAME, the first measurement system to collect real-time, in-the-wild data on modern CAPTCHA attacks. In order to do this, we first study the recent evolution in the protocols of CAPTCHAs as well as human-driven farms that facilitate attacks against CAPTCHAs. This study leads us directly to the discovery of a unique vantage point to conduct a global-scale CAPTCHA attack measurement study. Harnessing this, we design and build C-FRAME to be CAPTCHA-agnostic and ethically considerate. We then deploy our system for a 92-day period resulting in capturing of 425,257 CAPTCHA attacks on 1417 sites.

In order to characterize these attacks, we leverage a carefully designed qualitative analysis approach. Our study results in delineation of 34 different CAPTCHA-attack categories with several interesting real world attack examples. Twitter received the largest number of CAPTCHA attacks overall most of which attempt to create bot accounts. We also categorized and captured 34 attacks such as ticket scalping attempts (e.g. a Taylor Swift concert event in Brazil), fraudulent lawsuit claims, and abusive appointment booking attempts (e.g. a Spain visa site in China). We ascribe our attacks to 58 different countries across 5 continents. We present a detailed measurement analysis to give insights on this attack data and also suggest some future potential remediation measures that can be inspired by our system.

Acknowledgements

We would like to thank the anonymous reviewers and our shepherd, for their constructive comments and suggestions on how to improve this paper. We would like to thank David Pace and Vassil Roussev from University of New Orleans and Ali Ghosn from Louisiana State University for helping support the deployment of C-FRAME. Finally, we would like to thank Tharani Maaneeivaannan for first investigating the feasiblity of this measurement study. This material is based in part upon work supported by the National Science Foundation under grants: CNS-2126641 and CNS-2126655.

References

[1] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford, "CAPTCHA: using hard AI problems for security," in *Advances in Cryptology - EUROCRYPT 2003*.

[2] E. Bursztein, S. Bethard, C. Fabry, J. C. Mitchell, and D. Jurafsky, "How good are humans at solving captchas? A large scale evaluation," in *31st IEEE Symposium on Security and Privacy, SP 2010*.

[3] J. Yan and A. S. E. Ahmad, "Usability of captchas or usability issues in CAPTCHA design," in *Proceedings of the 4th Symposium on Usable Privacy and Security, SOUPS 2008, Pittsburgh, Pennsylvania*, ser. ACM International Conference Proceeding Series, 2008.

[4] A. Searles, Y. Nakatsuka, E. Ozturk, A. Paverd, G. Tsudik, and A. Enkoji, "An empirical study & evaluation of modern captchas," in *32nd USENIX Security Symposium 2023*.

[5] V. Shet, "Google security blog: Are you a robot? introducing "no captcha recaptcha"," https://web.archive.org/web/20160506184026/https://security.googleblog.com/2014/12/are-you-robot-introducing-no-captcha.html.

[6] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G. M. Voelker, and S. Savage, "Re: Captchas-understanding captcha-solving services in an economic context," in *19th USENIX Security Symposium, Washington, DC, USA, August 11-13, 2010, Proceedings*, 2010.

[7] J. Tam, J. Simsa, S. Hyde, and L. von Ahn, "Breaking audio captchas," in *Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems*, 2008.

[8] H. Gao, W. Wang, J. Qi, X. Wang, X. Liu, and J. Yan, "The robustness of hollow captchas," in *2013 ACM SIGSAC Conference on Computer and Communications Security*.

[9] E. Bursztein, J. Aigrain, A. Moscicki, and J. C. Mitchell, "The end is nigh: Generic solving of text-based captchas," in *8th USENIX Workshop on Offensive Technologies, WOOT '14, San Diego, CA, USA, August 19, 2014*, 2014.

[10] H. Meutzner, V. Nguyen, T. Holz, and D. Kolossa, "Using automatic speech recognition for attacking acoustic captchas: the trade-off between usability and security," in *Proceedings of the 30th Annual Computer Security Applications Conference, ACSAC 2014*.

[11] L. Tung, "Google algorithm busts CAPTCHA with 99.8 percent accuracy," https://www.zdnet.com/article/google-algorithm-busts-captcha-with-99-8-percent-accuracy/.

[12] H. Gao, J. Yan, F. Cao, Z. Zhang, L. Lei, M. Tang, P. Zhang, X. Zhou, X. Wang, and J. Li, "A simple generic attack on text captchas," in *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*, 2016.

[13] S. Solanki, G. Krishnan, V. Sampath, and J. Polakis, "In (cyber)space bots can hear you speak: Breaking audio captchas using OTS speech recognition," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec@CCS 2017*.

[14] K. Bock, D. Patel, G. Hughey, and D. Levin, "uncaptcha: A low-resource defeat of recaptcha's audio challenge," in *11th USENIX Workshop on Offensive Technologies, WOOT 2017, Vancouver, BC, Canada, August 14-15, 2017*, 2017.

[15] G. Ye, Z. Tang, D. Fang, Z. Zhu, Y. Feng, P. Xu, X. Chen, and Z. Wang, "Yet another text captcha solver: A generative adversarial network based approach," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018*.

[16] V. Shet, "Google security blog: recaptcha just got easier (but only if you're human)," https://web.archive.org/web/20160308094521mp_/https://googleonlinesecurity.blogspot.com/2013/10/recaptcha-just-got-easier-but-only-if.html.

[17] T. C. Blog, "Announcing turnstile, a user-friendly, privacy-preserving alternative to captcha," https://blog.cloudflare.com/turnstile-private-captcha-alternative/.

[18] Google, "reCAPTCHA Developer Guide," https://developers.google.com/recaptcha/docs/loading.

[19] Anti-captcha, https://anti-captcha.com/apidoc/task-types/RecaptchaV2Task.

[20] kolotibablo, https://kolotibablo.com/main/home.

[21] 2captcha, https://2captcha.com/api-docs/recaptcha-v2.

[22] D. B. Captcha, https://deathbycaptcha.com/api/newtokenrecaptcha.

[23] ImageTyperZ, https://www.imagetyperz.com/.

[24] 9kw, https://www.9kw.eu/api.html#apisubmit-tab.

[25] EndCaptcha, https://www.endcaptcha.com/api-specs#docrecaptchav2.

[26] 2captcha, https://2captcha.com/blog/bypassing-recaptcha-v2-on-google-search.

[27] anticaptcha, https://web.archive.org/web/20230528231822/https://anti-captcha.com/apidoc/task-types/AntiBotCookieTask.

[28] Arkoselabs, "Arkose Bot Manager Documentation," https://developer.arkoselabs.com/docs/verify-api-v4-response-fields.

[29] X. Lin, P. Ilia, S. Solanki, and J. Polakis, "Phish in sheep's clothing: Exploring the authentication pitfalls of browser fingerprinting," in *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, 2022.

[30] K. F. Team, https://www.facebook.com/kolotibabloofficial/photos/a.503691919787447/1673222479501046/?type=3.

[31] 2captcha, "Earn by promoting 2captcha," https://web.archive.org/web/20230329134221/https://2captcha.com/affiliate.

[32] M. Mohamed, N. Sachdeva, M. Georgescu, S. Gao, N. Saxena, C. Zhang, P. Kumaraguru, P. C. van Oorschot, and W. Chen, "A three-way investigation of a game-captcha: automated attacks, relay attacks and usability," in *9th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '14, Kyoto, Japan - June 03 - 06, 2014*, 2014.

[33] X. Mi, X. Feng, X. Liao, B. Liu, X. Wang, F. Qian, Z. Li, S. A. Alrwais, L. Sun, and Y. Liu, "Resident evil: Understanding residential IP proxy as a dark service," in *IEEE Symposium on Security and Privacy, SP 2019*.

[34] mitmproxy, https://mitmproxy.org.

[35] K. Ruth, A. Fass, C. Sadowski, E. Thomas, J. Azose, K. Ruth, M. Pearson, and Z. Durumeric, "A world wide view of browsing the world wide web," in *Internet Measurement Conference (IMC)*.

[36] S. Zhang, S. Pearman, L. Bauer, and N. Christin, "Why people (don't) use password managers effectively," in *Fifteenth Symposium on Usable Privacy and Security, SOUPS 2019, Santa Clara, CA, USA, August 11-13, 2019*, 2019.

[37] C. W. Munyendo, Y. Acar, and A. J. Aviv, ""desperate times call for desperate measures": User concerns with mobile loan apps in kenya," in *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*, 2022.

[38] J. M. Corbin and A. Strauss, "Grounded theory research: Procedures, canons, and evaluative criteria," *Qualitative sociology*, 1990.

[39] https://github.com/fakeshadow/psn_agent/blob/master/api.js.

[40] https://github.com/y95847frank/AutomatedTicketBot.

[41] https://github.com/xtekky/Twitch-Account-Gen.

[42] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html.

[43] K. Krippendorff, "Estimating the reliability, systematic error and random error of interval data," *Educational and psychological measurement*, 1970.

[44] K. De Swert, "Calculating inter-coder reliability in media content analysis using krippendorff's alpha," *Center for Politics and Communication*, vol. 15, pp. 1–15, 2012.

[45] https://en.wikipedia.org/wiki/Better_Online_Tickets_Sales_Act.

[46] https://www.taylorswift.com/tour/.

[47] https://twitter.com/t4f/status/1670537720760860673?s=20.

[48] https://x.com/t4f/status/1673063095457947648?s=20.

[49] https://www.nbcnews.com/news/latino/taylor-swift-law-introduced-brazil-scalpers-overwhelm-eras-tour-sales-rcna90519.

[50] https://www.coldplay.com/tour/.

[51] https://www.rugbyworldcup.com/2023/matches.

[52] https://www.billboard.com/pro/spotify-removed-thousands-songs-fight-streaming-fraud-ai/.

[53] https://support.tunecore.com/hc/en-us/articles/360056164472-What-is-Streaming-Fraud-Abnormal-Streaming-Activity-.

[54] https://www.musicbusinessworldwide.com/streaming-fraud-accounts-for-at-least-1-3-of-plays-on-services-like-spotify-and-deezer-in-france-shows-investigation/.

[55] https://www.investopedia.com/terms/c/classaction.asp.

[56] https://www.refererheadersettlement.com.

[57] https://mwpfsettlement.com.

[58] https://www.ftc.gov/enforcement/refunds/fashion-nova-settlement.

[59] https://www.ridester.com/amazon-flex-bot/.

[60] https://www.cnbc.com/2020/02/09/amazon-flex-drivers-use-bots-to-get-more-work.html.

[61] https://www.reseller.co.nz/article/698573/agile-robot-armies-batter-immigration-new-zealand-new-visa-system/.

[62] https://www.theguardian.com/uk-news/2023/sep/03/an-absolute-mess-learner-drivers-forced-to-buy-tests-on-black-market-as-companies-block-book-slots.

[63] https://despatch.blog.gov.uk/2023/06/29/how-were-dealing-with-bots-and-the-reselling-of-driving-tests/.

[64] https://www.kucoin.com/earn/x-lockdrop.

[65] https://www.change.org/p/to-approve-triumphfx-proportional-coin-settlement-proposal-with-immediate-effect.

[66] B. Acharya, M. Saad, A. E. Cinà, L. Schönherr, H. D. Nguyen, A. Oest, P. Vadrevu, and T. Holz, "Conning the crypto conman: End-to-end analysis of cryptocurrency-based technical support scams," in *2024 IEEE Symposium on Security and Privacy (SP)*, 2024.

[67] D. Dagon, C. C. Zou, and W. Lee, "Modeling botnet propagation using time zones," in *Proceedings of the Network and Distributed System Security Symposium, NDSS 2006, San Diego, California, USA*.

[68] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the botnet phenomenon," in *Proceedings of the 6th ACM SIGCOMM Internet Measurement Conference, IMC 2006, Rio de Janeriro, Brazil, October 25-27, 2006*, 2006.

[69] P. A. Porras and H. Saïdi, "A foray into conficker's logic and rendezvous points," in *2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats, LEET '09, Boston, MA, USA, April 21, 2009*.

[70] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. A. Kemmerer, C. Kruegel, and G. Vigna, "Your botnet is my botnet: analysis of a botnet takeover," in *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*, 2009.

[71] B. Stone-Gross, T. Holz, G. Stringhini, and G. Vigna, "The underground economy of spam: A botmaster's perspective of coordinating large-scale spam campaigns," in *4th USENIX Workshop on Large-Scale Exploits and Emergent Threats, LEET '11, Boston, MA, USA, March 29, 2011*, 2011.

[72] D. Y. Wang, S. Savage, and G. M. Voelker, "Juice: A longitudinal study of an SEO botnet," in *20th Annual Network and Distributed System Security Symposium 2013*.

[73] M. Antonakakis, T. April, M. D. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in *26th USENIX Security Symposium, USENIX Security 2017*.

[74] P. Tan and V. Kumar, "Discovery of web robot sessions based on their navigational patterns," *Data Min. Knowl. Discov.*

[75] A. Lourenço and O. Belo, "Catching web crawlers in the act," in *Proceedings of the 6th International Conference on Web Engineering, ICWE 2006, Palo Alto, California, USA, July 11-14, 2006*, 2006.

[76] M. Allman, V. Paxson, and J. Terrell, "A brief history of scanning," in *Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference, IMC 2007*.

[77] J. Lee, S. D. Cha, D. Lee, and H. Lee, "Classification of web robots: An empirical study based on over one billion requests," *Computers and Security*, vol. 28, no. 8, 2009.

[78] A. Balla, A. Stassopoulou, and M. D. Dikaiakos, "Real-time web crawler detection," in *18th International Conference on Telecommunications, ICT 2011, Ayia Napa, Cyprus, May 8-11, 2011*, 2011.

[79] K. Thomas, D. McCoy, C. Grier, A. Kolcz, and V. Paxson, "Trafficking fraudulent accounts: The role of the underground market in twitter spam and abuse," in *Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA, August 14-16, 2013*, 2013.

[80] S. T. Jan, Q. Hao, T. Hu, J. Pu, S. Oswal, G. Wang, and B. Viswanath, "Throwing darts in the dark? detecting bots with limited data using neural data augmentation," in *2020 IEEE symposium on security and privacy (SP)*. IEEE, 2020, pp. 1190–1206.

[81] D. Canali and D. Balzarotti, "Behind the scenes of online attacks: an analysis of exploitation behaviors on the web," in *20th Annual Network & Distributed System Security Symposium 2013*.

[82] X. Li, B. A. Azad, A. Rahmati, and N. Nikiforakis, "Good bot, bad bot: Characterizing automated browsing activity," in *2021 IEEE symposium on security and privacy (sp)*. IEEE, 2021.

[83] B. Kondracki, J. So, and N. Nikiforakis, "Uninvited guests: Analyzing the identity and behavior of certificate transparency bots," in *31st USENIX Security Symposium*.

[84] S. Sebastián, R. Diugan, J. Caballero, I. Sánchez-Rola, and L. Bilge, "Domain and website attribution beyond WHOIS," in *Annual Computer Security Applications Conference, ACSAC 2023, Austin, TX, USA, December 4-8, 2023*. ACM, 2023, pp. 124–137. [Online]. Available: https://doi.org/10.1145/3627106.3627190

Appendix A. Responsible Disclosure

The gravity of our measurement results prompted video meetings with heads of research/product teams from three

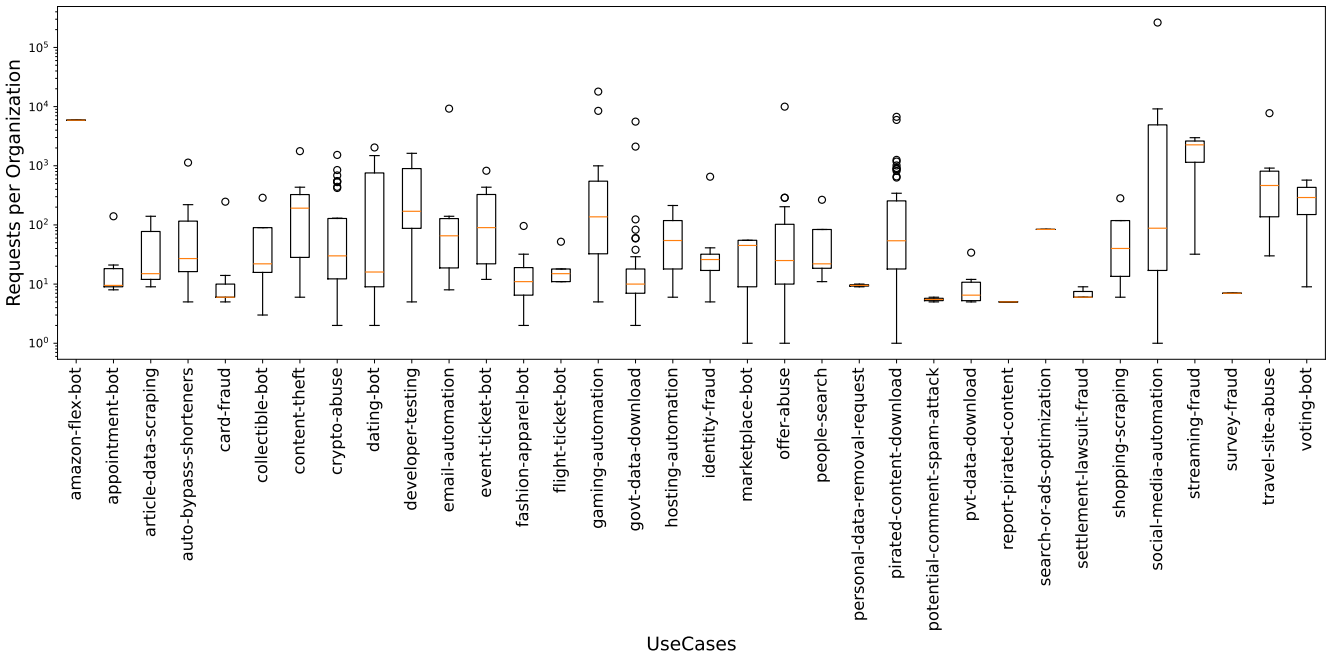

Figure 6: Distribution of requests across multiple orgs for the 34 attack categories

highly popular CAPTCHA-service providers upon our disclosure to them. All three teams expressed interest in our research as it sheds light on the attacks being experienced by other CAPTCHA services. One team was interested in future collaborative defensive approaches (as described next) by harnessing C-FRAME's data in real-time for providing high-quality ground truth for ML-based abuse detection models.

While this initial disclosure effort was positive, we recognized the possibility that the services might not be able to relay fine-grained attack information to the affected clients. There, we have made detailed disclosure reports to the top ten targeted orgs. Among them, two teams have escalated our reports to their technical teams and have followed up on technical discussion about our measurement approaches and results. We have also shared all attacks in the "Survey fraud" category with the research teams that are conducting the surveys. Among them, one research group responded saying that their online survey hosted on Qualtrics was entirely stalled immediately due to the onslaught of bots, thus validating our findings. They were appreciative of our findings and responded as follows: "I think your work on this is incredibly important and necessary". They also mentioned that they decided to unfortunately transition away from online surveys due to the frequent bot interference issue.

As making such disclosure reports manually to all affected targets is time-intensive, we then utilized a recently released domain attribution tool [84] that leverages WHOIS data, passive DNS and TLS certificate information to find contact information for all 1023 affected orgs. With the help of this tool, we were able to send initial contact e-mails to 130 orgs out of which about 15 orgs have responded asking for more details. One of these orgs is a popular news organization in the US whose IT team met with us to discuss collaborative solutions for the problem of "bot newsletter signups" that they are currently facing.

Domain	# Requested Days
twitter.com	87
sony.com	86
linkedin.com	84
live.com	84
roblox.com	84
filer.net	82
expedia.com	82
turbobit.net	81
expedia.es	80
ea.com	80

TABLE 6: Top 10 domains that are repeatedly targeted on multiple days

Appendix B. Recommendations

Apart from providing insights into characterizing bot attacks, C-FRAME can also be used as a tool for remediating bot attacks on specific sites and services.

For Target Websites Websites can leverage C-FRAME to discern if they are under bot-targeted attacks and precisely identify the specific URLs facing threats. The insights gained empower website administrators to make targeted adjustments. As demonstrated in earlier sections, a minimal number of accounts is sufficient to comprehensively capture and analyze a diverse array of attacks. In such cases, sites grappling with bot attacks can gain more benefits from C-FRAME by adopting single-use URLs for CAPTCHA-enabled

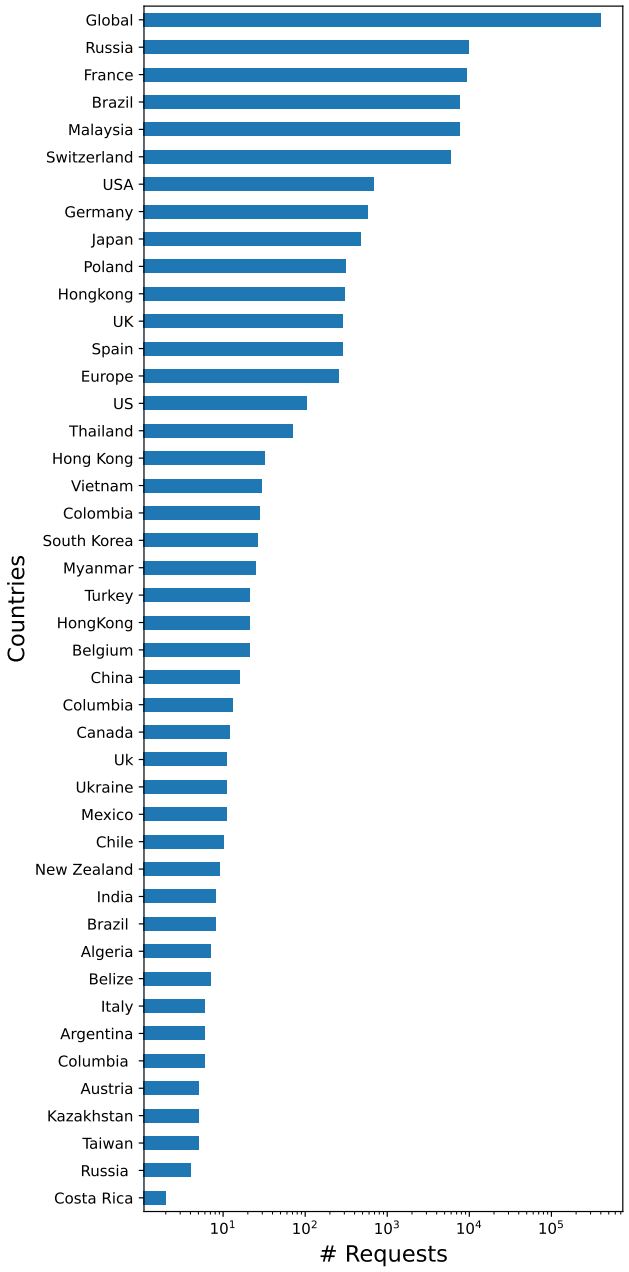

Figure 7: Total attack requests by country

pages where each URL on which a CAPTCHA is presented includes a random value and is thus used only one time. This practice facilitates unique identification of attack requests based on just the URL and timestamp thus allowing for efficient fingerprinting of attackers. Cross-referencing these fingerprints with other server requests helps the sites to go beyond the data captured in C-FRAME. However, in this context, it is essential to acknowledge a potential evasion attempt in the future by CAPTCHA farms, which may omit the URL path from being forwarded to the farms, thus diminishing the efficacy of single-use URLs.

Org	# Requests	# ccTLDs	Use Case
twitter	255,480	1	social-media-automation
live	18,383	1	social-media-automation
roblox	17,930	1	gaming-automation
openai	9980	1	offer-abuse
sony	8472	1	gaming-automation
expedia	7743	8	travel-site-abuse
astro	7503	1	unknown
turbobit	6715	1	pirated-content-download
linkedin	5995	1	social-media-automation
filer	5953	1	pirated-content-download
amazon	5892	1	amazon-flex-bot
pbh	5574	1	govt-data-download
snapchat	4552	1	social-media-automation
onlyfans	3532	1	content-theft
deezer	2982	1	streaming-fraud
spotify	2258	1	streaming-fraud
fazenda	2108	1	govt-data-download
ourtime	2037	1	dating-bot
mysite	1623	1	developer-testing
claimbits	1525	1	crypto-abuse

TABLE 7: # requests for top org

Alternatively, browser/device fingerprinting and analysis of IP addresses, often available from CAPTCHA services like Arkose Labs, provide an additional layer of defense. This combined approach significantly expands the scope of identifying and mitigating bot-related requests. Integrating single-use URLs with fingerprinting and IP address analysis enables websites to fortify their defenses against evolving evasion tactics by sophisticated bot attackers. Regular updates of such gained forensic intelligence can ensure a proactive stance against emerging bot threats.

For CAPTCHA Services. Similar to our recommendations for targeted websites, CAPTCHA services can implement C-FRAME to establish a definitive understanding of requests originating from bots. By running C-FRAME in-house, CAPTCHA services can generate immediately trackable token strings in response to CAPTCHA challenge requests. These tokens are then provided to attackers, who must submit them to the webserver to bypass the CAPTCHAs. The web server subsequently verifies the tokens with the CAPTCHA service. Once verified, the CAPTCHA service notifies the web server, enabling it to fingerprint the web bot traffic. Furthermore, this approach will help the CAPTCHA services gain valuable and definitive ground truth on the behavior of farm workers when solving the CAPTCHAs which can ultimately be used to improve their behavior-based farm and bot-traffic detection algorithms.

Appendix C.
Meta-Review

The following meta-review was prepared by the program committee for the 2024 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

C.1. Summary

The paper describes the first study that characterizes the attacks on CAPTCHAs in the wild. The authors study the evolution of CAPTCHAs by studying 6 present CAPTCHA APIs, as well as how adversaries have evolved alongwith, by studying the APIs of 6 CAPTCHA-farms. Using a key insight derived from this initial study, the authors developed C-FRAME, a measurement system for analyzing attacks on CAPTCHAs in the wild which mimics CAPTCHA farm workers. Deploying C-FRAME for 92 days, the authors collect over 425k CAPTCHA attacks on 1418 sites, which they analyze both qualitatively and quantitatively.

C.2. Scientific Contributions

- Provides a New Data Set For Public Use
- Creates a New Tool to Enable Future Science
- Provides a Valuable Step Forward in an Established Field

C.3. Reasons for Acceptance

1) The evaluation uncovers several key insights that challenge long-held assumptions (e.g., the fact that attackers don't send the CAPTCHA frame by frame to a farm, but instead, receive the puzzle from the services themselves.
2) The approach is more ethical than prior work, as it does not contribute to the problem or lead to unintended financing of CAPTCHA-farms, as it simply drops challenges by modifying the site key.
3) The qualitative analysis of CAPTCHA attacks uncovers evidence of several types of criminal activity, such as settlement and ticketing fraud, cryptocurrency abuse, visa appointment abuse, and spam.

C.4. Noteworthy Concerns

1) The paper studies only two farms. As a result, the measurements and findings may not generalize.