Domains Do Change Their Spots: Quantifying Potential Abuse of Residual Trust

Johnny So, Najmeh Miramirkhani, Michael Ferdman, Nick Nikiforakis
Stony Brook University

***Abstract*—When domains expire and are released to the public, adversaries can re-register them with the hope of exploiting residual trust from clients that are unaware of the change in ownership. Because domain name resolution is integral to the web, possible clients run the gamut from humans browsing the web to automated and periodic processes such as system updates. For an adversary, this trivially yields access to an attack vector that can affect a multitude of diverse systems and devices. We reason that some domains which experience residual trust and are valuable from a security perspective are not valuable from a dropcatching perspective and, as such, can be re-registered by an adversary without participating in fiercely competitive auctions for expired domains.**

In this paper, we present an investigation into this attack vector using a *top-down*, opportunistic approach, as opposed to *bottom-up* approaches used by prior work that target specific systems and infrastructure. Throughout a one-month re-registration period, we identify potentially valuable dropped domains using a threshold of passive DNS resolutions, re-register, and deploy them with basic honeypot services. We then analyze the traffic to these domains to find instances of residual trust that can be exploited. Our honeypot services recorded, over a four-month period, 650,737,621 requests from 5,540,379 unique IP addresses situated in 22,744 different autonomous systems to the 201 re-registered domains. Although a majority of these domains may not pose significant security risks, we are most concerned with and thus focus our discussion on unusual domains which receive orders of magnitude more traffic and types of traffic that are significantly different from the other domains. These include domains which previously functioned as a torrent tracker, an API for a computer lab usage statistics service used by many universities, an API that was a point of contact for a common Android haptics library, security company DNS sinkhole servers, an Internet radio and music station, command-and-control servers for malware and potentially unwanted programs, and an email tracker. Our findings demonstrate that expired domains pose a real threat to the security of the Internet ecosystem and that adversaries with modest budgets can compromise a wide range of systems and services by merely registering previously-popular domains that were left to expire.

I. Introduction

The Domain Name System (DNS) is the distributed phone book of the Internet, tying human-readable domain names to their actual IP address(es). Because DNS is integral to the web experience, DNS queries are ever-present, yet abstracted away from the normal user experience. Queries are performed in the background and are answered by optimized, local, and dedicated DNS resolvers that find the IP address for a given domain name in milliseconds. Unsurprisingly, details such as the domain registration process and the registration life cycles are not important to users.

Perhaps the most pressing yet unimportant detail is that domains can expire and change ownership without the users' knowledge. If a domain expires and changes ownership, its residual trust — the historical reputation — is implicitly transferred to its new owners [1]. There are no standards or built-in mechanisms that warn clients when a previously contacted domain is now operated by a different entity. This enables an attack surface that exploits prior trust and is a subset of the overall problem of integrity verification on the web. If a malicious actor were to re-register an expired domain with residual trust, they would be able to exploit all residual trust traffic. All clients that depend on domain name resolution are susceptible to contributing to residual trust traffic, and potential clients run the gamut from end users, to JavaScript libraries, to system updates.

There has been a rising interest in expired domains; industry interest took shape in the rise of the dropcatch ecosystem in which companies build complex infrastructure to re-register valuable expired domain names as soon as they are released to the public. The dynamics behind this phenomenon and its resultant industry were studied by previous works [2], [3], [4]. The value of such domains is typically derived from characteristics such as their string length, previous amount of traffic, popularity ranking in lists such as Alexa's Top 1 Million, and past indexing in search engines.

On the other hand, academic interest took shape in investigating how expired domains can enable an attack on what may be an otherwise secure system. For example, Lever et al. have found instances of residual-trust issues in domain names used by browser extensions [1], as well as malware and potentially unwanted programs [5]. Alowaisheq et al. discovered that even dangerous domains that were sinkholed by security researchers and governments are eventually allowed to expire and can therefore be re-registered and used to revive dormant botnets [6]. Other researchers have pointed out the interaction between web security mechanisms and expired domains, showing that the latter can be abused to inject malicious JavaScript in popular domains [7] as well as evade previously secure CSP policies [8]. In all these cases, the abuse of residual trust in these domains would compromise the security of entire systems and potentially harm the privacy of a significant portion of their user base.

By taking a step back, we observe that prior work from the academic community favored a *bottom-up* approach, in which they began with analyzing a target system (such as extensions, malware, and CSP policies) and then discovered and quantified how expired domains can compromise them. In this paper, we present the first *top-down* approach to investigate this problem — we re-register domains in a target-agnostic fashion *and then* not only detect but also quantify the potential abuse of residual trust in our pool of domains. Our approach resembles that of an opportunistic attacker who re-registers a number of expired domains and discovers that some of them still receive traffic because of residual trust, whereas a bottom-up approach first identifies systems of interest and then attempts to re-register particular domains in hopes of exploiting residual trust. Using our top-down approach, we find that not only are the specific worries described by these bottom-up works likely to occur in reality, but they are also not financially difficult to successfully execute.

Our study seeks to understand and quantify the feasibility for valuable domains to slip by the dropcatch industry and be obtained by lower-budget registrants. With the ever-increasing number of domain registrations [9] and consequently ever-increasing number of domain expirations, we argue that a better understanding of the

feasibility and severity of this attack vector is necessary to ensure the security of not only end users but also dependent systems. In particular, there are domains that would not be considered valuable from a typical monetary or dropcatching perspective, but may hold significant value from a security perspective. Domains that do not serve end-user-facing content have no need for a memorable name nor do they appear in common domain ranking lists. Some common examples include domains for DNS name servers, API servers, or CDN servers. To this end, we design an experiment to simulate the process of a potential malicious actor aiming to re-register expired domains whose residual trust can be exploited to cause harm to the security and privacy of dependent systems and end users.

We summarize our contributions as follows:

- **Large-scale analysis and profiling of residual trust traffic to 201 domains over four months.** We empirically demonstrate that it is likely for a new owner to identify the type of service previously offered on a particular domain using not only residual traffic logs, but also third-party references on the Internet and historical archives. Using this method, we were able to confidently categorize 128 domains from our pool of re-registered domains. Additionally, we find evidence that residual trust traffic is not guaranteed to decay over time; that is, it is possible to continue to receive traffic because of residual trust for *months* after a domain experiences a change in ownership.
- **Design, implementation, and evaluation of an infrastructure to aid in identifying domains with exploitable residual trust traffic.** Our top-down approach comprises automated pipelines for the selection, re-registration, and deployment of expired domain candidates as well as for the detection of residual trust traffic in our domains.
- **Evidence that residual trust abuse can affect millions of different IP addresses over tens of thousands of different autonomous systems, even with a simple domain selection strategy, and it does not require heavy financial investment.** We spend a total of $1,464.64 to re-register 201 domains and, after profiling the re-registered domains, find that some would enable an adversary to not only gather information but also manipulate and compromise unaware clients and systems. These include notable, high-volume domains which previously functioned as a torrent tracker, an API for a computer-lab-usage statistics service used by many universities, an API that was a point of contact for a common Android haptics library, security company DNS sinkhole servers, an Internet radio and music station, command-and-control centers for malware and potentially unwanted programs (PUPs), and an email tracker [10].

II. Residual Trust in DNS

Information about domains is managed by registrars and registries who follow standardized processes created by the Internet Corporation for Assigned Names and Numbers (ICANN). Depending on the top-level domain (TLD) and the corresponding managing entity, such processes may vary. For example, country-code TLDs are managed by their respective countries, whereas generic TLDs are managed by ICANN. Domains are generally registered for several years, after which they expire and must be renewed. Generic TLDs follow ICANN's Expired Registration Recovery Policy (ERRP) [11], a process which grants expiring domain owners buffer time to renew their domain before it is released to the public. The ERRP mandates that domain registrars attempt to notify domain owners twice (up to one month) before expiration and once (within five days) after expiration. After the registrar deletes a domain, it enters the Redemption Grace Period, during which the expired domain owner can still renew the domain, although typically at a higher price. The expired domain becomes publicly available for re-registration five days after the grace period ends.

As there is no inherent mechanism to warn of changes in domain ownership, it is likely that unaware clients continue to attempt to contact domains after expiration and after their deletion by their registrar. If a deleted domain is purchased and receives such residual traffic, the new owner will have access to all incoming communications from these clients. In this scenario, traditional techniques such as HTTPS will not maintain confidentiality. In fact, any encryption technique that derives a key through communication with the server will not maintain the confidentiality of the data, since the receiver is operated by the re-registrant. Perhaps the only communications that might remain opaque to a re-registrant are obscure or non-standard methods which may be nontrivial to reverse engineer.

It is important to acknowledge that there have been remote identity verification mechanisms that were proposed such as HTTP Public Key Pinning (HPKP) [12]. HPKP would be able to notice that the identity of the server has changed because of a change in the server's public key, but HPKP is now deprecated [13] because of its difficulty to use and risks such as rendering a site unusable and hostile pinning [14]. Even if HPKP were enabled for the domain, it would be difficult to determine whether a pinning violation was caused by a misconfiguration or an attack. It would also be difficult to determine if the new domain owner is malicious and attempting to exploit residual trust or benign and merely offering new content on the domain.

As the new owner is privy to all communication to their domain, they would be able to identify the type of service previously offered by the domain and build specific infrastructure in an attempt to gather more information from higher levels of interaction with clients. Depending on the nature of the services previously offered by the domain, the new owner can gain varying levels of personally identifiable information, including IP addresses, geolocation, device fingerprints, and user credentials.

To the best of our knowledge, this study is the first to use a top-down approach to investigate the feasibility of residual trust abuse. We design and implement pipelines to automatically identify potentially valuable domains, re-register and deploy them, and identify cases of residual trust. We adopt a domain selection strategy that relies on passive DNS, which refers to the capturing of live DNS records, consequently building partial replicas of DNS zones, and storing them in a database [15]. Passive DNS data help to answer questions that are difficult or impossible with standard DNS, such as historical resolution information (e.g., "What IP address did this domain name point to in the past?").

III. Experiment Design & Methodology

In this section, we present the rationale behind our experimental design, beginning with our strategy for domain selection and automated infrastructure for re-registration, deployment, and monitoring. We end with detailing the methodology of residual trust detection in our traffic analysis.

A. Domain Selection

Our domain selection criterion is derived from the intuition that domains with a large amount of traffic pre-expiration are more likely to receive traffic post-expiration because of their sheer

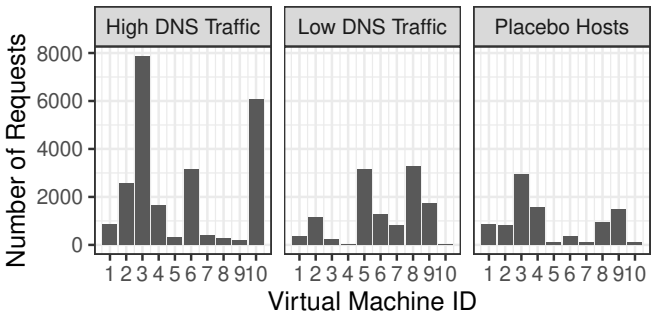

Fig. 1. Traffic received on recently expired and re-registered domains. Each virtual machine was assigned a unique IP address. Placebo hosts are control hosts that do not have an associated domain name.

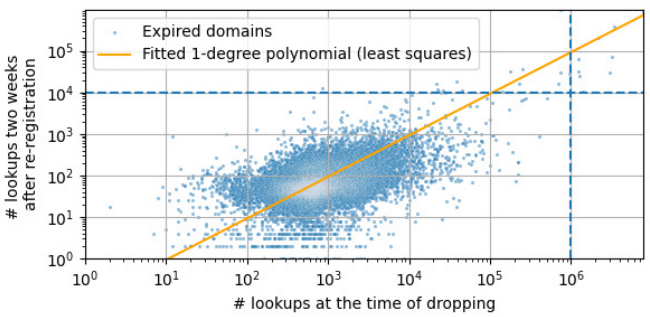

Fig. 2. Pre-expiration vs. post re-registration DNS resolutions of a random sample of re-registered dropped domains. After re-registration, the number of DNS resolutions is generally lower by one order of magnitude. All sampled domains with over 1 million pre-expiration resolutions receive at least over 10K resolutions after re-registration.

popularity, whether the traffic is from unaware users or to-be-updated infrastructure tools. The number of DNS resolutions for a domain is an appropriate proxy for the amount of traffic received by a domain, because DNS is one of the fundamental services on the Internet and the protocol is necessary to resolve domain names to IP addresses. We use a commercial passive DNS database [16] as our proxy to gauge DNS activity for a given domain and attempt to re-register domains with a high number of pre-expiration resolutions.

Because dropcatchers are known to re-register domain names as soon as they become available and may spend hundreds of thousands of dollars for that privilege [3], [4], we focus on domain names that are of no interest to them. Namely, the threat model that we investigate is that of an attacker who can opportunistically identify the "hidden gems" that dropcatchers missed (i.e., ones that were not registered immediately after being dropped) and can register them for nominal prices (i.e., the typical price of a domain registration). This attacker will register domains based on how many times they used to be resolved by DNS when they were active, as opposed to looking for the same traits that dropcatchers look for (e.g., their historical Alexa ranking, how short each domain is, if there's a website available for it on Internet Archive, etc.) [4].

We first conducted two small-scale monitoring experiments before our domain registration process: the first to determine whether the intuition that valuable domains expire and receive residual trust traffic is reasonable and the second to determine an appropriate empirical threshold for the number of historical DNS resolutions to use for our domain selection. In the former, we use the passive DNS database and re-register ten historically high-traffic domains several hours after they were released for public registration. These domains were selected by sorting a sample of 20K domains that were dropped on June 13, 2019 by the number of their historical DNS resolutions. Because these domains remained on the market for several hours after they became available, we know that dropcatchers were not interested in them; according to Lauinger et al., dropcatchers register their target domains mere seconds after they are released [3]. As part of our control groups, we also re-registered ten historically low-traffic domains, and created ten *placebo* machines that are not associated with any domain names. These machines receive traffic only from network-scanning bots that probe online IPs. All machines in the three groups logged the timestamp and client IP address of each HTTP(S) request and responded with a blank page. In addition, to ensure that there is no difference in network placement, we place all the groups in the same class-C subnet.

Figure 1 shows the number of requests received by each container in a one-week period. We observe that the low-traffic group and the placebo group received similar volume of HTTP(S) traffic; these low-traffic domains did not receive more traffic than any other online container, regardless of whether their IP resolved from a registered domain. On the other hand, we observed that two historically high-volume domains received more than twice as much traffic as any historically low-volume or placebo container. This suggests that the domain names which were not deemed valuable by dropcatchers are still referenced by third-party, dependent infrastructure that is unaware of the change in ownership.

In our second motivating experiment, we obtained a list of dropped domains on June 20, 2019 and monitored the domains to select those that were re-registered within two weeks. This sample includes a mix of domains that were dropcaught and domains that were re-registered later. For each selected domain, we compared its total number of DNS resolutions (accounting for SOA, A, NS, and MX records) at the time of its expiration with its number of additional DNS resolutions two weeks after re-registration. Figure 2 compares the number of lookups after re-registration with the number before expiration, showing that the number of resolutions after re-registration is generally one order of magnitude lower than the number of resolutions pre-expiration. We empirically choose a threshold of 1 million resolutions pre-expiration which is a likely indicator that the domain will receive over 10K resolutions within two weeks after re-registration. This experiment also served as an estimate of the amount of traffic that our re-registered domains would receive. We restrict the TLDs considered to the most popular (`.com`, `.net`, `.org`, `.info` and `.biz`) to narrow down the set of domain candidates. Although the TLDs we consider are popular among dropcatchers, our domain selection strategy is different from that of typical dropcatchers as described by Miramirkhani et al. [4], which reduced the contention for the domains that we selected.

Our domain registration process consisted of a pilot phase and a main phase. In our pilot phase, we experimented with domain selection strategies and re-registered 29 domains; in our main phase, we re-registered 172 domains over the course of one month beginning on August 8, 2019. For the first 20 days, we randomly sampled the dropped domains to reduce the number of queries to the passive DNS database, due to service quotas. In the last 10 days, we were able to query all dropped domains and identify more candidates. On several occasions, our daily candidate identification process resulted in sets

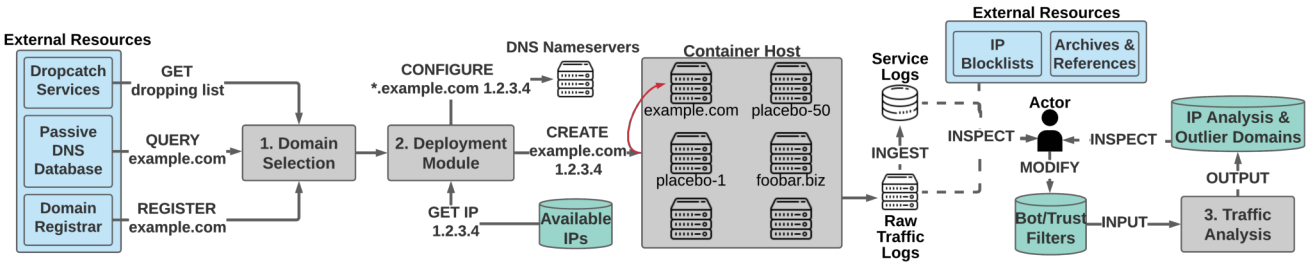

Fig. 3. A diagram of our system modules. Our deployment module also installs honeypot services for each registered domain and all logs, including raw packet capture files, are collected by our raw log storage. All the dashed lines on the right-hand side belong to an iterative process involving manual analysis.

of domains with similar lexical structures (e.g., `vvol1kans.com`, `vvol1ccan.com`, `wol1ck.com`, `wol1kkano.com`, and `vvol1can.com`). These domains used squatting techniques that were more complex than typical techniques [17], [18], [19], [20], [21] and were difficult to automatically detect; for this reason, we manually filtered out these similar domains. For each set of similar domains, we opted to keep the one with the highest amount of traffic. In rare cases, if we were able to find some interesting characteristics of a lower-volume domain, we would select it instead of the highest-volume domain.

Our main registration phase identified 550 domains of interest; of these, 321 domains (58% of all identified) were manually identified as highly similar to another candidate and excluded from consideration. We were unable to register 57 domains (24% of the remaining candidates) because they were re-registered by another entity. It is unclear whether these domains were immediately caught or re-registered hours after they were dropped because our module executed at regular intervals. We were able to successfully re-register 172 (75%) of the remaining candidates.

B. Infrastructure

Figure 3 presents a high-level overview of our system, comprising a domain selection module, container and honeypot deployment module, log collector, and traffic analyzer. The domain selection module performed daily queries to Domainmonster [22], DynaDot [23], NameJet [24], Pool [25], and SnapNames [26] to obtain a list of domains that were about to expire and followed the process outlined in the previous section to select the domains considered valuable enough to attempt to re-register.

Upon a successful re-registration of the domain name, the deployment module creates a new container on the host machine, allocates it a unique IP address (from our range of two class C blocks), and creates the corresponding DNS records in our two name servers. We use wildcard DNS (NS, A, and MX) records to capture requests to all subdomains to quantify the effects of residual trust as accurately as possible. In addition to the containers for re-registered domains, we also randomly create containers that are not tied to any domain name, but are otherwise identically configured and located in the same IP address range. Following our terminology from Section III-A, we call these extra containers *placebos* and use them to filter out traffic to our domain servers that is not caused by residual trust. We do not register domains for these placebo hosts to minimize our costs and because we had already observed that they receive similar amounts of traffic as historically low-traffic domains from our initial experiment. Because these servers are not tied to any domain name, clients can only discover them through network scanning. We reason that these network-probing clients will also contact our domain servers, enabling us to use traffic to our placebo servers as a type of negative filter in our analysis of residual trust traffic (we further elaborate on how we accomplish this in Section III-D).

We configure the containers with well understood and popular protocols on the Internet, as the protocols that will actually receive residual trust traffic will vary depending not only on the specific domains registered, but also on the time of re-registration for these domains. Moreover, traffic on non-standardized ports may be associated with more than one service which would make it challenging to interpret incoming traffic to these ports on the fly. As a result, the deployment module configures containers to listen for traffic on only the most common service ports, namely HTTP(S), SSH, Telnet, and FTP. All HTTP(S) requests are served a custom 404 error page with *bot traps* (see Section III-D), along with a script that attempts to collect a browser fingerprint. HTTPS is supported using an X.509 certificate from Let's Encrypt [27]. The SSH service is a medium-interaction honeypot that logs a large selection of possible events [28]. The Telnet and FTP services are low-interaction and collect only the credentials of the login attempts [29].

In addition to the honeypot service logs, we collect packet captures of the first 64 bytes of every packet entering the containers. The logs are regularly collected from each container by a log-collector module and stored on a large network-attached storage device. Our traffic analyzer then ingests all raw logs from the log collector into an Elasticsearch database [30].

C. Residual Trust Detection

Public systems on the Internet are bound to receive large amounts of automated traffic, whether they are from spam, benign crawlers for search engines and historical archives, or malicious fingerprinting tools and vulnerability scanners. Identifying such automated traffic is a prerequisite to our further analysis of residual trust traffic. As a first step in our analysis, we therefore distinguish between two types of accesses, `bot` traffic and `trust` traffic, using architectural elements, request characteristics, and IP blocklists.

Our traffic analysis pipeline combines the intuition behind the common block-listing and allow-listing approaches. The pipeline comprises a series of `bot` filters followed by `trust` filters that query our database. If a request matches a specific filter, the corresponding client IP address is tagged with the filter type and name. An IP address may be associated with multiple `bot` filters and `trust` filters, but `bot` filters (the block-lists) are assigned more weight than `trust` filters (the allow-lists) — any IP address with

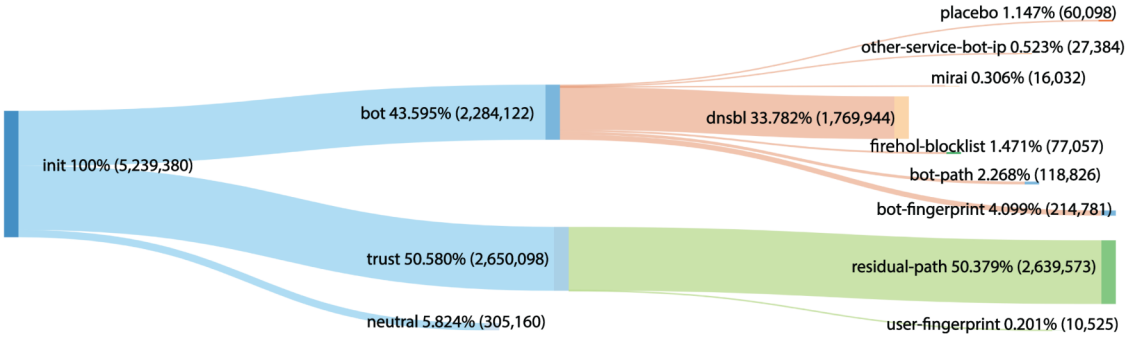

Fig. 4. Sankey diagram of the results of the bot and residual trust detection pipeline on HTTP(S) traffic. Each flow represents an exclusive set of IP addresses and each node a stage in the filtering process. Each IP is bucketed in the first matching filter flow and the filters were applied sequentially to all client IP addresses in the top-down order depicted, for each horizontal depth level from the initial node. For example, if an IP is tagged with bot filters of {`dnsbl, bot-fingerprint`} and a trust filter of `residual-path`, the IP address would be in the flow [`init, bot, dnsbl`] because `bot` precedes `trust` and `dnsbl` precedes `bot-fingerprint`. IP addresses are marked `neutral` if they have neither filter tag applied (i.e., they did not exhibit any indication that their requests resulted from `bot` or `trust` traffic). Section III-D includes a detailed discussion of the analysis pipeline.

a mix of both types will be marked `bot`. Thus, an IP address will be characterized as *a*) `bot` if it has any `bot` filter tags, *b*) `trust` if it has no `bot` filter tags and any `trust` filter tags, or *c*) `neutral` if it has no associated filter tags.

We categorize clients at IP address granularity. Although an IP address can be shared by multiple clients, it is difficult to determine the client or session boundaries for a given IP address from a server-side perspective. Using heuristics (e.g., consider all requests from the same IP address within 60 seconds of one another to be one session) is prone to false positives in bot classification. We instead opt for a broader, IP address granularity, and note that any number of IP addresses is a lower bound of the actual number of clients behind those IPs. For classifying residual trust traffic, this approach is conservative — if any of the accesses from an IP address are classified as `bot`, all requests from that IP address are considered as `bot`. In most cases, we present the residual trust traffic results as a percentage of IP addresses tagged `trust`.

D. Bot and Trust Indicators

We develop a traffic analysis pipeline which comprises a series of filters. The filters rely on architectural elements, suspicious request characteristics, and external resources. Architectural elements are intrinsically part of our system: the placebo servers, bot traps, and a JavaScript fingerprinting script. Most of the other filters are derived from suspicious request characteristics (e.g., resembles an exploit or fingerprint attempt) that were manually identified for each type of service. Our last type of filter relies on information gleaned from external resources, such as web archives and IP blocklists used for traditional networking rules. All filters are run sequentially and the results of this filter for HTTP(S) traffic can be seen in Figure 4.

a) Architectural elements (placebos, bot traps, and fingerprints): Placebo servers are servers that are not associated with any expired domain names, but are otherwise identical in configuration. They are allocated at random IP addresses from the same range and run the same set of services as our domain servers. Because the placebo servers are not associated with any of the domains we re-register, all traffic they receive is not related to residual trust from DNS. We therefore use them as a control group to learn the network signatures of bot traffic, and we mark all IP addresses that access any service on any placebo hosts as bots. The corresponding tag that represents this filter in Figure 4 is `placebo`.

Our second and third architectural mechanisms affect only the HTTP(S) service, with which we serve a web page that includes "bot traps" and an innocuously-named fingerprinting script (`utils.js`). We use the popular FingerprintJS [31] fingerprinting script, extended with a function to also send a POST request to a public IP geolocation service. Bot traps are designed to lure bots into requesting certain paths that would never be requested by an end user web browser [10], [32]. Every trap is associated with a unique URL in its `href` attribute or in its content, and these trap elements include *a*) inconspicuous 1x1 images, *b*) HTML comments, *c*) dynamically removed content after the page is loaded using JavaScript, *d*) elements that are not displayed (i.e., `display=none`), and *e*) elements that inherit the invisible property from their parents.

Crawlers can fall for all of the traps or none of them, depending on their crawling logic. However, prior work has shown that bot behavior is generally too simple to avoid all of these traps [33]. Even sophisticated crawlers, such as those used for research or by search engines like Google, still fall for these traps. Moreover, we discovered that 4,764 unique IP addresses that self-identified as bots in the HTTP User-Agent header fall for these bot traps (approximately 43.42% of all IPs that self-declare themselves as a bot). We tag any IP addresses that request trap paths as bots and the corresponding tag in Figure 4 is `bot-path`.

Our last architectural mechanism is an augmented fingerprinting script [31], which also collects the document referrer and sends an extra POST request for the current IP address geolocation before sending the final fingerprint in the payload of a POST request back to the container. The lack of a fingerprint can be used to identify bots, because most do not support JavaScript [33]. Although some crawlers support JavaScript, the benign ones can be easily identified by the `User-Agent` header and by their well-known IP address ranges. If these values are spoofed (such that one is pretending to be a crawler), in our results, it will constitute a false negative rather than a false positive of an instance of residual trust (which is still aligned with our goal of quantifying the lower bound on the amount of residual trust traffic). We use the fingerprints for multiple filters:

- for each fingerprint, if the `User-Agent` contains "bot" or a reverse DNS lookup on the IP address yields a domain that

Listing 1. Filter to detect and tag IP addresses for repeated login attempts. `d1` and `d2` are Levenshtein distances. `bot_like_pw` checks whether the password contains a common variant of the domain name and whether the password is similar to one from RockYou's leak.

```
def tag_repeated_logins(ip, d1=1, d2=2, t=120):
    is_bot = False
    is_user = False
    for r1 in get_reqs_from_ip(ip):
        is_bot = bot_like_pw(r1.domain, r1.pw, d2)
        if is_bot:
            break
        for r2 in get_similar_reqs(ip, r1, d1, t):
            is_bot = bot_like_pw(r2.domain, r2.pw, d2)
            if is_bot:
                break
        is_user = len(s_reqs) > 0
        if is_user:
            break
    tag_ip(ip, is_bot, is_user)
```

contains popular search engine keywords (e.g., "google"), the IP address will be tagged as a bot

- for each request from a major browser (detected using the `User-Agent` header [34]), mark the IP address with `bot` or `trust` depending on whether there is a corresponding fingerprint for the request

In our dataset, we observe approximately 1M fingerprints from 52K unique IP addresses. The corresponding tag that represents these filters in Figure 4 are `bot-fingerprint` and `user-fingerprint`.

b) Request characteristics: Our second major type of traffic analysis filters rely on suspicious-request characteristics for each service. For our HTTP(S) service, the filters focus on request paths; for the credentials-based services (SSH, Telnet, and FTP), the filters focus on username and password combinations.

Our bot filters for web traffic identify attempts to *a*) use exploits (e.g., a path traversal attack [35] or a CVE for a web service), *b*) access a backdoor (e.g., `/shell.php`), *c*) fingerprint the web service [33] (e.g., common paths used by BlindElephant [36]), *d*) access backups files (e.g., `/wp-config.old`), and *e*) make a login request with HTTP method `POST`. The content that we serve does not include a `POST` login form, so these have to be automated, credential-stuffing requests. The corresponding IP addresses are tagged with `bot-path` in Figure 4.

We also manually categorized the services offered by each domain on the HTTP(S) protocol, pre-expiration, and assign a confidence level of *low* or *high*. We rely on, in decreasing order of priority, the Wayback Machine for historical archives, combing Google for references to the domain, and identifying request patterns in our logs to assign a category representing the type of service the domain previously offered. If we could not categorize a domain on the HTTP(S) protocol, we attempt to use the most popular port to categorize traffic. If the most popular port for such a domain is a standard port, we include a breakdown, otherwise we place the domain in the Unknown category.

To identify request patterns, we make use of our database's optimized query and aggregation functionality [30]. For each domain in our dataset, we start with a traffic sample of at most 3 requests per IP address per data shard [37] and find keywords in request paths that are significantly more popular within the particular domain as compared to across all domains [38]. Requests for paths that match these significantly more popular keywords within the domain are then aggregated for manual inspection.

TABLE I
HIGH-LEVEL TRAFFIC VOLUME STATISTICS TO HONEYPOT SERVICES IN DOMAIN SERVERS.

Service	# Requests	# Unique IPs	# ASNs
HTTP(S)	421M	5.2M	21K
SSH	195M	110K	5.4K
Telnet	34M	240K	8.6K
FTP	279K	6K	1.1K
Total	651M	5.5M	23K

In addition to a category label, each domain is also annotated with a confidence level of: *a*) *low* if we can only infer from third-party references or popular request paths, or *b*) *high* if we found an archived version of the domain no older than six months and the URL structure of the archived page is similar to the traffic patterns we observed.

For the domains with a *high* confidence label, we manually compile a list of *residual paths* for each domain: paths that used to exist on the domain pre-expiration. Through inspection of the links on an archived domain's homepage, we construct a per-domain set of residual path patterns. Main navigation links and other prominent endpoints were prioritized in the set for each domain. If many service-relevant endpoints were found to have the same request path prefix (e.g., `/a/b/c`, `/a/b/d`), we subsume them into a wildcard query record with the longest matching prefix (e.g., `/a/b/*`) to account for potential new paths that were added after the archival date. The residual path patterns for each domain also include traffic patterns observed in our logs that are of similar structure to the ones in the archived version. General paths that are common to many domains and are not service-specific are excluded (e.g., `/contact`). Client IP addresses that request any of these residual paths are marked with the `trust` filter `residual-path` in Figure 4.

For our credentials-based services, we rely on credential patterns to identify suspicious requests. Our main filters match requests with the following characteristics:

- **(Bot)** passwords in the 1,000 most popular passwords of the RockYou database breach [39],
- **(Bot)** passwords that contain the domain name [33],
- **(Bot)** username and password combinations that are also used in traffic to our placebo servers. There is a potential for false negatives if there are legitimate username and password combinations that have been guessed by bots.
- **(Trust)** multiple login attempts with credentials that are nearly the same (max Levenshtein distance of 1 for username and for password) within a window of 120 seconds. Listing 1 includes a more detailed description of the algorithm. We reason that bots are more likely to attempt logins from credentials lists, as opposed to brute-force guessing, and these credentials are a farther distance apart. Repeatedly attempting the same or very similar credentials yields less potential gain and can be an indication of human behavior (i.e., trying to log in again thinking there was a typo in the password).

Our last filter that uses request characteristics identifies suspicious requests from a packet structure perspective instead of a particular service's perspective. We base this filter on prior network monitoring efforts which identified a characteristic of some Mirai variants that send TCP SYN probes with the TCP sequence number set to the desti-

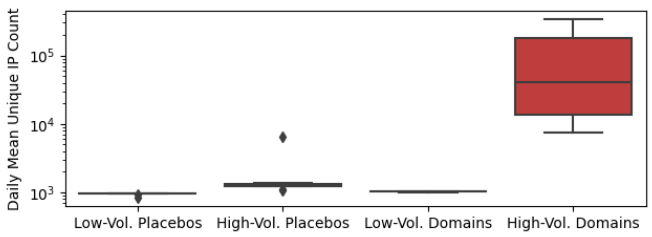

Fig. 5. Number of unique IP addresses that contact the ten least and most popular placebo servers and expired domains daily. The popularity of each server was ranked using the total number of unique IP addresses in its traffic.

nation IP address when displayed byte-by-byte in dotted-decimal notation (e.g., `0x48c1af41` for `72.193.175.65`) [40]. We scan our TCP header capture files and mark the IP addresses that match this condition as bots. The corresponding tag in Figure 4 is `mirai`.

c) IP blocklists: IP blocklists are our sole external bot detection mechanism. Blocklists are well-established in industry and are commonly used for routing and firewall rules [41]. We use multiple DNS blocklist (DNSBL) providers and IP blocklists related to reputation and cyber crime as aggregated and categorized by FireHOL [42]. DNS blocklists were created for and typically used to protect mail servers against spam mail. Although our primary purpose is not to detect which IP addresses send spam mail, it is likely that any traffic coming from these addresses are not due to residual trust. We use 52 DNSBL providers including Spamhaus [43], SORBS [44], and Barracuda Central [45].

IV. Traffic Analysis

In this section, we report our findings on the data collected from 201 re-registered domains in the time period from August 1, 2019 to December 1, 2019. Our honeypot services collected a total of 650,737,621 requests to our re-registered domain servers from 5,540,379 unique IP addresses distributed among 22,744 unique autonomous systems; see Table I for a breakdown of the number of requests, number of unique IP addresses, and number of autonomous systems for each service. We begin by conducting a high-level verification of our filtering mechanisms and then characterizing the traffic volume and port popularity of our servers using packet-level and honeypot service log-level perspectives. We then present the results of using the bot and trust indicators described in Section III-D and detail the high-level temporal characteristics of the residual trust traffic for select domains.

A. High-Level Verification of Filtering Mechanisms

Because we conducted our experiment in the wild, we do not have access to a ground-truth dataset. The closest data point to ground truth could be the fingerprinting script that was served over HTTP(S), but this would not capture some valid use cases (e.g., if a domain used to respond to requests from automated tools that have no need to support JavaScript). We opted to perform a high-level verification of our results by considering the blocklist filters (DNSBLs and FireHOL's IP blocklists) as a ground truth, comparing the result of using the other filtering mechanisms with the result of using only the blocklist filters.

The blocklist filters marked 1,866,840 IP addresses as suspicious (bot-like); all other filtering mechanisms marked 417,282 IP addresses as suspicious. There are 332,513 IP addresses (79.68% of 417,282) that were flagged suspicious by both the blocklist filters and another

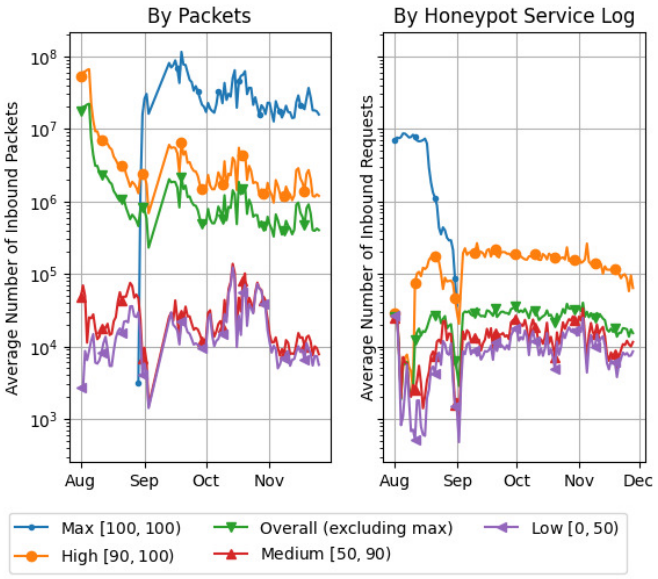

Fig. 6. Traffic volume levels for all domains. Bucket labels were assigned based on either the total number of inbound packets or of honeypot service log requests. The corresponding value for a bucket on a day is the mean of all its domains' value for that day. The intervals in each label refer to percentile threshold cutoffs. The low traffic bucket comprises up to the 50th percentile and medium up to the 90th percentile. Note the 100th percentile domain is different depending on the grouping strategy.

filter. This suggests that most IP addresses which exhibit suspicious behavior are likely to already be on a blocklist and that the filtering mechanisms confirm one another. However, the non-blocklist filters have their own merit in identifying IP addresses that exhibit suspicious behavior without the need to resort to a trusted third party (i.e., the provider of a blocklist).

B. Characterizing Volume and Popularity

a) Stratified traffic volume: Figure 5 shows the distribution of the number of IP addresses that contact the ten most and least popular placebo servers and domains. We compare the same number of servers in each group because we deployed fewer placebo servers (52) than domain servers (201). We find that the less popular domains receive traffic from a similar number of IP addresses as our placebo hosts, but the more popular domains receive substantially more traffic, more than an order of magnitude more IP addresses compared to the most popular placebo servers.

Beyond the large difference in traffic volume between the placebo servers and domain servers, we also see a pronounced difference in traffic volume within our domain servers. Figure 6 visualizes the underlying distribution of traffic to our domain servers by grouping domains depending on both their aggregate packet count and honeypot service log count using the 50th and 90th percentiles as thresholds. The packet-level grouping reveals that there is a vast amount of traffic that is not captured by our service logs — the 100th percentile server in the packet-level grouping is not the same as the 100th percentile server in the log-level grouping because of vast amounts of traffic to ports without running services. For the packet-level grouping, the highest-volume server is `tianxingmeng.com` which received significant traffic to the ports {8000,6600}, whereas, for the log-level grouping, `ipv6tracker.org` received the largest number of service requests as it previously hosted a torrent tracker on port 80.

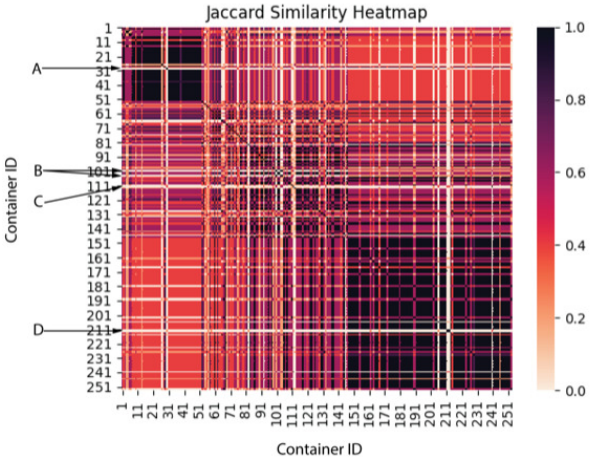

Fig. 7. Similarity heatmap of the five most popular inbound ports (by number of packets received) for all 253 containers (placebos and domains) in our experiment. Axis labels correspond to an ID number for each container, temporally ordered by re-registration date. All placebo containers appear on the axes before containers associated with re-registered domains (thus comparisons between placebo containers are in the top-left corner).

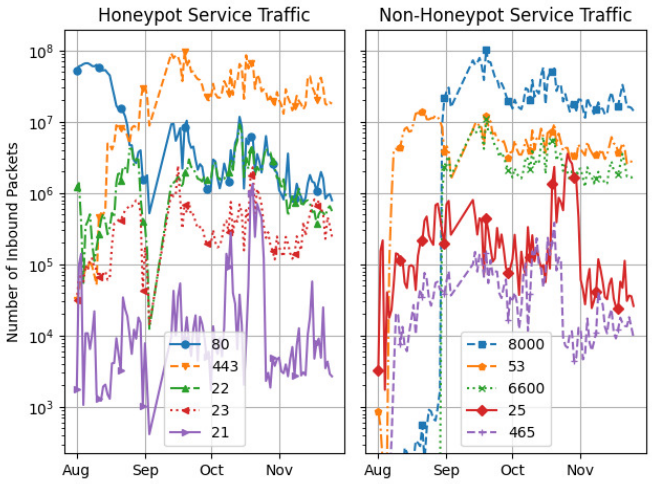

Fig. 8. Traffic volume for the ten most popular ports to all domains.

We further elaborate on the characteristics of the traffic to these two domains in Section V.

b) Dissimilar types of traffic: We characterize the type of traffic received by each server as an ordered tuple of their five most popular inbound ports (by raw packet count) and disregard the magnitudes. Hence, using this representation, we compare only the port numbers and the order of the most popular ports, and not the number of packets sent to each port. Figure 7 summarizes this in the form of a heatmap whose values are weighted Jaccard similarity scores of the five most popular inbound ports for two servers. A high score (dark color) means that the five most popular inbound ports for one server are similar to the five most popular inbound ports for another server.

From the dark square region in the upper-left of the heatmap, we find that the majority of placebo servers receive traffic to the same ports. There are notable outliers whose most popular ports are radically different from the rest of the servers and they are visualized as white rows and columns. Interestingly, we see an outlier placebo server at **A**. Whereas the most popular ports for placebo servers were typically $[22, 80, 23, 25, 2222]$, **A**'s most popular ports were $[23, 22, 445, 80, 8089]$. We attribute this phenomenon to a one-off cause as it is unique among all placebos and there would be no reason for a network scanner to probe only the particular IP address for **A** and not others in the network. The other outliers are all from our domain servers. **B** corresponds to two domains which previously functioned as sinkhole DNS name servers for a security company, and **C** corresponds to another high-volume DNS server. **D** corresponds to two domains that received large amounts of traffic to ports 8000 and 6600. We take a closer look at the traffic to these outliers in Section V.

c) Port popularity: Requests sent to ports that do not have an associated service listening on them are bound to not receive a reply. Thus, honeypot services on expired domains might increase port popularity only if the clients responsible for the traffic value meaningful replies (i.e. expect replies to be in a certain format). We find that the lack of interaction and associated service for non-honeypot ports does not deter certain clients from contacting some of our domains, as can be seen in Figure 8. Many of these are worrisome; we find that the five most popular non-honeypot ports include both standardized ports for DNS and SMTP and non-standardized but popular ports for alternative HTTP and Internet radio/music [46], [47]. Owning a DNS name server enables controlling all clients who depend on the server for domain resolutions. Mail server owners are able to eavesdrop on email communications, regardless if the communication is encrypted with TLS. Controllers of popular radio and music servers can cherrypick content that will be absorbed by their listeners. We make these examples concrete in Section V by discussing particular domains that offered these services.

We also note that there is a distinct popularity ordering among our honeypot services, with HTTP(S) as the most popular and followed by SSH, Telnet, and FTP, and suspect that the lack of depth in interaction provided by our credentials-based honeypot services might have contributed to this phenomenon. Thus, we present the results of our residual trust detection pipeline on, mainly, the HTTP(S) service traffic in the next section.

d) Profiling domains: We report our findings from our manual domain service profiling for the 201 re-registered domains in detail in Table II. Of the domains selected by our strategy, approximately 20.90% were previously gambling websites that used real world currency and 16.42% were related to crime, either offering contrabands, document forgery, or serving as a command-and-control server.

Many of these services can be abused by an adversary who obtains ownership of the corresponding domain, particularly if they are high volume. A malicious actor who closely monitors traffic to each domain could identify, profile, and then straightforwardly categorize a domain with residual trust traffic. With a general idea of what the domain previously offered, they would be able to better tailor their own honeypot services to gather and exfiltrate personally identifiable information.

C. Pipeline Results for Honeypot Services

We broadly distinguish the HTTP(S) protocol from the credentials-based services because of the difference in interaction and available information for each request. Although we serve the same 404 page to all HTTP(S) requests, the requests allow for a higher level of interaction (and consequently more log information) than the credentials-based services that only collect login attempts. Clients

TABLE II
HTTP(S) SERVICE PROFILING BREAKDOWN FOR EVERY DOMAIN. PORT **H** REPRESENTS HTTP AND HTTP(S) ON PORTS 80 AND 443 AND NON-**H** POPULAR PORTS INCLUDE {23,25,1443,8000,6600}. TRAFFIC VOLUME LEVELS WERE ASSIGNED BASED ON THE 50^{TH} AND 90^{TH} PERCENTILE OF TOTAL PACKET COUNTS TO EACH DOMAIN SERVER.

Category **Port**	Conf.		Volume Bucket			Total
	Low	Conf.	L	M	H	
Gambling **H**	3	39	25	16	1	42
Crime **H**	5	30	20	12	3	33
Streaming **H**	3	12	6	9	0	15
Adult **H**	0	12	6	6	0	12
Company **H**	0	8	2	3	3	8
DNS **53**	0	5	0	0	5	5
Non-**H**	0	4	1	1	2	4
Downloads **H**	0	4	3	0	1	4
API **H**	0	3	0	0	3	3
Email **25**	0	1	0	0	1	1
Other **H**	0	13	5	7	1	13
Unknown **-**	60	-	32	26	2	60
Total	73	128	100	80	21	201

making a request for a specific URL on HTTP(S) will expect to receive certain content and, if this expectation is not met, they may attempt a request for another URL. With each request, we are able to log HTTP headers of interest and potentially collect an associated browser fingerprint. Thus, our pipeline for HTTP(S) traffic is more intricate and has more filters than our pipeline for the credentials-based services.

In fact, because our credentials-based services collect minimal information, it is hard to distinguish `trust` traffic from `bot` traffic for these services. For SSH and Telnet traffic, we find that no IP addresses were marked `trust`; for FTP traffic, we find that one IP address (0.017%) was marked `trust` with the repeated credentials filter. This IP address sent two requests to one of our domains from a Google IP address; we can conclude with high confidence that this case is a false positive because the domain was not affiliated with Google. As such, we focus our discussion on our findings from the HTTP(S) traffic.

Most of our domains did not receive substantial `trust` traffic, as shown in Figure 9. The two cumulative distribution functions of the percentage of IP addresses marked `bot` or `trust` are symmetrical. Notably, although the majority of our re-registered domains have little potential for abuse of residual trust, they are not the ones that we are concerned with. Rather, the outliers that lie at the extremes of each curve are of most interest. These are high-volume domains with traffic originating from millions of IP addresses. We summarize the impact of the domains with the highest trust-related HTTP(S) traffic in Table III.

Figure 4 details the breakdown of the effects of each `bot` and `trust` indicator from Section III-D. Each node in the diagram represents a stage in the filtering process and its corresponding set of unique IP addresses, and the width of each flow from node A to B represents the number of unique IP addresses in A that were filtered by B. The sets of IP addresses represented by each node are exclusive and the top-down order of nodes describes the unique criteria satisfied by those IP addresses. In summary, we find that at least 50.58% of the IP addresses in our HTTP(S) traffic are unaware of changes in domain ownership and send residual trust traffic to our servers. In contrast, approximately 43.59% of the IP addresses exhibit `bot`-like behavior and 5.82% could not be classified as either `bot` or `trust` using our indicators.

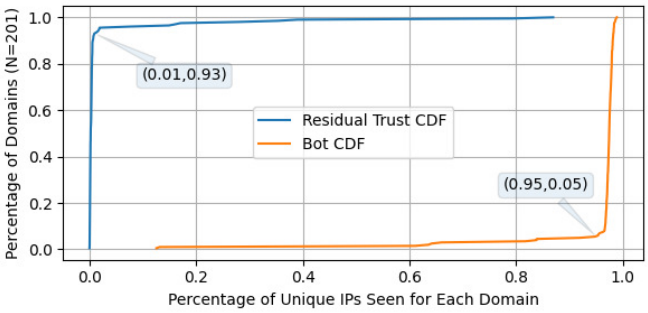

Fig. 9. CDF of the percentage of bot and trust HTTP(S) traffic for each domain in terms of unique number of IP addresses. Over 95% of domains received over 95% bot traffic, and only 7% of domains received over 1% trust traffic.

D. Temporal Characteristics

Figure 10 visualizes both the daily number of unique IPs that contact domains on high-volume, non-honeypot ports on the left and the daily number of unique `trust`-tagged IP addresses that visit domains with a large percentage of trust-related traffic on the right. From the left-hand plot, it appears that the number of IPs that attempt to contact a non-honeypot service on these domains largely remains constant over our analysis time window. This suggests that residual trust traffic is not guaranteed to decay over time, or, at least, not within a period of a few months after an expired domain is re-registered.

As for the right-hand side, we see evidence for both sides, namely that residual trust traffic may largely remain stable or that it may gradually decay. Domains `labstats.go`, `avantmobile.com` and `facecommute.com` exposed APIs which were accessed by automated and unaware clients and exhibited stable trust-related traffic. This implies that the clients who attempt to contact these domains have not been updated. Similarly, domain `tianxingmeng.com` likely hosted a radio and music streaming station and now also experiences stable trust-related traffic after re-registration. The other two domains, `cpttrk.com` and `parastrok.info`, previously hosted an advertising service and a sports news website and they both experience volatile trust-related traffic. In particular, we see that the curve for `parastrok.info`, which previously offered news articles, dies off less than a month after re-registration of the domain. The curve for `cpttrk.com` also decreases, but gradually over time and experiences large and sudden spikes.

Both plots suggest that residual trust traffic is not guaranteed to decay over time. Depending on what the domain was previously used for and the clients that contact it, it may be difficult to reconfigure the clients to use another point of contact. On the other hand, we also see evidence that residual trust traffic could decay over time. In fact, we see that in some cases it may suddenly drop, indicating an update or reconfiguration in the clients, or gradually decay over time with a volatile instantaneous behavior. In conclusion, although we see that the potential for residual trust abuse may be mitigated over time, we cannot rely on this phenomenon because it is not guaranteed.

V. CASE STUDIES

In this section, we present several case studies from our pool of registered domains, demonstrating the magnitude and wide range of abuse possible via the mere re-registration of previously-popular domains. Table III presents the statistics of the HTTP(S) traffic for

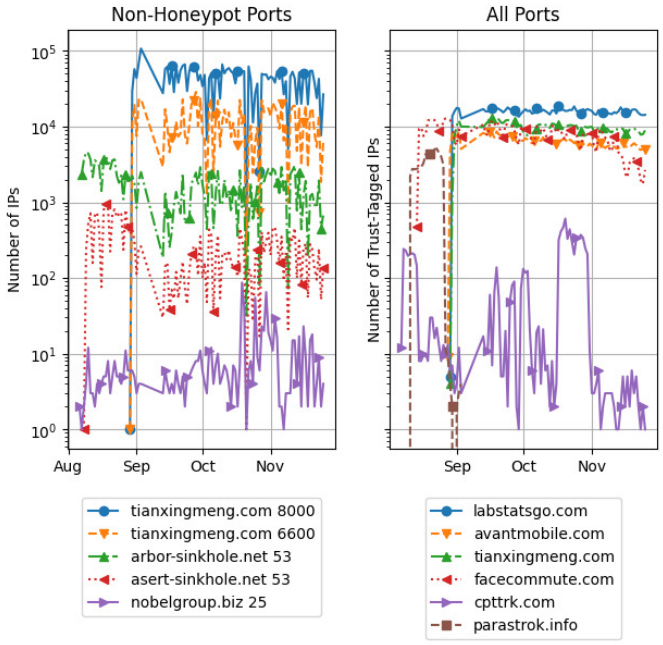

Fig. 10. Temporal (in)stability of trust-related traffic. The left side plots the daily number of unique IP addresses that contact domains on high-volume, non-honeypot ports. The right plots the daily number of unique `trust`-tagged IP addresses from HTTP(S) analysis that contact domains with a large percentage of trust-related traffic on all ports.

the top ten domains with the highest percentage of `trust`-tagged IP addresses, while Table IV shows the statistics in non-honeypot service traffic to high-volume domains.

a) APIs: `ipv6tracker.org` previously functioned as a popular torrent tracker, a component that aids peers in discovering one another. Torrent trackers maintain a list of peers that expressed interest in a given torrent, enabling the tracker to inform new clients about the peers it can connect to. This expired domain received the most traffic to honeypot services in our dataset, and it was beyond our expectations and ability to efficiently handle. We only retain data for a one-month window for this domain, and we note from Figure 6 that its presence led to a system-wide outage of our infrastructure near the beginning of September. Because we did not expect significant new findings from this domain, we opted to leave it offline to limit further resource use and log space consumption. We observe a daily average of approximately 7M requests before the container started to experience availability problems. Almost all requests are to `/announce`, `/announce33`, and `/scrape` on host `tracker.ipv6tracker.org` with query keys such as `info_hash`, `peer_id`, `port`, `uploaded`, `downloaded`, and `left` and the User-Agent header set to a torrent client (e.g., uTorrent, BitTorrent, and qTorrent). This trivially grants the ability to gather information and create a database of files seeded and downloaded by all the peers in the network. Controlling such a popular torrent tracker would expose a large threat vector that would affect over 2.5M unique IP addresses situated in over 9.4K ASNs.

Other noteworthy domains that exposed an API are `labstats.go`, `fgmail2.com`, and `avantmobile.com`. `labstats.go` was previously an API domain for *Labstats*, a tool that tracks time, location, and method of student usage of computer lab resources, trusted by over a thousand colleges according to their website [48]. Residual trust traffic requests are for the paths `/Assign/ServerSlot` and `/Api/Client`. Control of this domain would allow for large-scale exfiltration of data from higher education facilities. In a similar fashion, owning `fgmail.com` would also yield valuable information because it previously functioned as an email tracker. We know from prior work that email tracking is privacy-harming and it is not uncommon for senders to intentionally leak email addresses to third-party trackers [49].

Control of `avantmobile.com` would allow for gathering of personally identifiable information on a much larger scale because it exposed an API for an Android libary. Profiling this domain was the result of combining different pieces of residual information. We began by noting that the majority of requests are HTTP POSTs to `/log` and `/sync` for the subdomain `api.avantmobile.com` with the User-Agent header set to a version of `okhttp`, a popular Android HTTP library. Using a Java decompiler to investigate archived versons of Android applications, we found that disparate apps that were on Google Play [50], [51], [52] used a common third-party haptics library, `com.immersion.content`, which contained string references to `api.avantmobile.com`. It is unclear whether this library was the original or a modified version as it is no longer offered by the publisher, but it appears to have been used by many Android apps. These apps are no longer available in the Google Play store, but an APK mirror self-reports over 500K downloads for one of them from their website [52]. We confirmed our suspicions when we captured DNS queries for `api.avantmobile.com` by running these mobile apps on an emulator and monitoring network traffic. Furthermore, the top 10 ASNs by the number of IPs observed in the traffic to the `avantmobile.com` domain are all ISPs that offer telecommunications services. The last mentions we found for the domain are social media accounts on Facebook [53] and Instagram [54] referring to an "Avant Mobile" company that lists the domain as their website, although we suspect this to be a front for malicious behavior.

In summary, obtaining control of high-volume APIs can be quite attractive to attackers. In the case of a torrent tracker, the new owner can learn which peers host which files, their locations, and attempt to attack connecting clients. In the case of a service trusted by many educational institutions, the new owner gains a method to attack many geographically diverse institutions trusted by the general populace. In the case of a suspicious domain contacted by a widely-used mobile application library, the new owner can gather highly confidential information about a vast array of people for further use.

b) Malicious activity: We discovered multiple domains in our dataset linked to malicious activity. Of this group, `facecommute.com` received the largest amount of traffic. According to some malware detection services, it previously functioned as a command-and-control server [55]. Common request paths to this domain include `/bots/{log,install-failure,update-additional-data}` and `/api/{poll,log}`. Query key parameters in the requests point to an entity named `cloudnet` (e.g., `cloudnet_{guid,file,process}`) and likely refer to their C2 entities hosted in the cloud. Another expired domain previously used for malicious purposes is `gbox-data.net`. According to VirusTotal, it is linked to a family of PUPs named Guardbox [56]. This domain was unique in that clients used subdomains as if they were request paths (i.e., clients would send a `GET /` request to `gb-alive-msg.gbox-data.net` instead of a `GET /gb-alive-msg` request to

TABLE III

SERVICE, VOLUME, AND TRUST STATISTICS IN HTTP(S) TRAFFIC TO THE TOP TEN DOMAINS WITH THE HIGHEST PERCENTAGE OF TRUST-TAGGED IPS. BUCKET RESULTS USING BOTH METRICS (RAW PACKET COUNT AND SERVICE LOG COUNT) FOR DOMAIN TRAFFIC VOLUME GROUPING FOR EACH DOMAIN AS DISCUSSED IN SECTION III ARE INCLUDED. THE LAST COLUMN DESCRIBES THE PERCENTAGE OF IP ADDRESSES THAT WERE TAGGED AS TRUST. SEE SECTION V FOR IN-DEPTH CHARACTERIZATIONS OF SOME OF THESE DOMAINS.

Domain	Service		Traffic Bucket		Traffic Statistics			Trust %
	Category	Description	Packets	Logs	Requests	IPs	ASNs	
labstats.go	API	Computer lab usage	High	High	144M	52K	1.5K	86.93%
ipv6tracker.org	Downloads	Torrent tracker	High	Max	138M	2.5M	9.4K	79.27%
avantmobile.com	API	Android haptics library	High	High	23M	683K	8.1K	38.95%
tianxingmeng.com	Non-HTTP	Internet radio & music	Max	High	121K	28K	1.1K	35.38%
facecommute.com	Crime	Command and control	High	High	55M	1.3M	6.0K	27.72%
cpttrk.com	Company	Advertising	High	High	886K	34K	2.8K	17.06%
parastrok.info	Other	News	High	High	13M	133K	6.1K	16.09%
fgmail2.com	Company	Email tracking	Medium	Medium	80K	9.2K	1.5K	14.92%
rgbdomsrv.com	Crime	Adware	High	High	12M	294K	5.3K	7.57%
tattooes.info	Other	Images	Medium	High	611K	16K	2.5K	1.88%

gbox-data.net). Obtaining control of domains associated with malicious activity and botnets would enable new owners to hijack these networks of compromised machines at low financial cost.

c) Name servers: DNS servers are crucial to the Internet. There have been many attacks on and misconfigurations of DNS infrastructure that paralyze regions of the web [57], [58]. Surprisingly, we were able to register *multiple* high volume name servers; see Table IV. Even more surprising, two of the name servers were sinkholes for a security company contracted by many ISPs to protect against DoS attacks [59]. Security sinkhole name servers are typically responsible for receiving traffic from infected machines attempting to resolve the domain of their main C&C server. By obtaining control of such servers, a malicious actor can potentially hijack all of the C&C botnets that security companies and registrars worked to redirect. We were also able to obtain control of internetemc.com, which belonged to an ISP in Georgia. It did not receive as many requests as one might expect for having belonged to an ISP, most likely because this domain was a secondary name server.

Re-registration of name servers is particularly critical, because it enables the new owner to attack all clients that rely on the name servers. Furthermore, the issue is compounded by traditional techniques such as load balancing. If an attacker gains control of a fraction of a load-balanced set of name servers, network administrators face more difficulty in diagnosing the problem because it manifests in non-deterministic behavior.

d) Stale content and resources: We obtained a class of expired domains that offered now-stale content and resources on the web. The most notable is tianxingmeng.com, which was our highest traffic domain in terms of raw packet count. Based on the massive number of packets to ports $\{8000, 6600\}$, it is likely that the domain previously hosted an internet radio stream [47], [46]. Obtaining control of such a domain enables the owner to manipulate its content however they wish. This, in turn, enables an adversary to more easily conduct social engineering attacks and cause social unrest through techniques such as fake news. Other domains in this category include ctnetload1.com, which served a JavaScript file (/js/?wkey=qKpVXT) used by cryptocurrency blogs including smartereum.com [60], forklog.media [61], and profitgid.ru [62], which we identified using the HTTP Referer header present in some requests and verified using the Wayback Machine. We found that forklog.media still included this stale script in February 2020 [61], over 6 months since we had re-registered the domain. Another notable domain in this class is tattooes.info, whose gallery of tattoo images was indexed by search engines including Google. Obtaining control of a domain that delivers content and resources to other domains can enable social engineering and supply chain attacks.

e) Mirrored domains: After profiling and categorizing each of the domains we registered, we found evidence of several domains belonging to the same entity. These mirrored domains resolved to the same IP address before they expired and were structurally similar, but not to the extent of typosquatting one another (e.g., leon4399.com and leon5044.com). In particular, we found mirrored domains for five different entities, four of which offered online gambling services and one offering a document forgery service. The largest gambling service entity created a new mirrored domain on a daily basis [63], which appears to be a strategy for evading domain blocklists.

Re-registering mirrored domains would enable an adversary to more easily perform phishing attacks. For example, if an adversary obtains an old mirror of an online service and deploys identical or similar content, a user may be misled to enter their credentials. The attacker could then use these credentials to authenticate with the actual domains used by the service and inflict financial harm or compromise other accounts belonging to the same user. The adversary could also manipulate the served content however they wish, potentially injecting malicious scripts or malware that will be downloaded by unsuspecting users. Moreover, because mirrored domains receive only a fraction of the entire user base, it may be difficult for service administrators to diagnose the problem.

VI. RELATED WORK

The security community has previously investigated the abuse of residual trust in expired domain names in a *bottom-up*, or targeted, approach. Prior work began with a target in mind and investigated how expired domains could be used to exploit residual trust traffic to and from these targets. For example, Moore and Clayton studied what became of expired US bank domains after such banks shut down, merged with other institutions, or in general stopped using their domains [64]. Lever et al. introduced the concept of residual trust and how it implicitly transfers to new owners [1]. The authors also observed the growing phenomenon of residual trust abuse and proposed a defense mechanism to locate potential changes in domain ownership. In later work, the authors discovered that expired domains

TABLE IV
SERVICE, VOLUME, AND TRUST STATISTICS IN NON-HONEYPOT SERVICE TRAFFIC TO HIGH-VOLUME DOMAINS. BUCKET RESULTS USING BOTH METRICS (RAW PACKET COUNT AND SERVICE LOG COUNT) FOR DOMAIN TRAFFIC VOLUME GROUPING FOR EACH DOMAIN AS DISCUSSED IN SECTION III ARE INCLUDED. SEE SECTION V FOR IN-DEPTH CHARACTERIZATIONS OF SOME OF THESE DOMAINS.

Domain	Service		Traffic Bucket		Traffic Statistics		
	Port	Description	Packets	Logs	Packets	IPs	ASNs
tianxingmeng.com	8000	Radio	Max	High	2.3B	1.45M	1.9K
icuesta.net	53	DNS	High	Low	359M	63.5K	8.0K
tianxingmeng.com	6600	Music	Max	High	247M	550K	1.4K
arbor-sinkhole.net	53	DNS	High	Low	123M	10.2K	619
internetemc.com	53	DNS	High	Medium	66M	26.2K	3.2K
abiling.com	53 00	DNS	High	Low	22M	10.5K	1.5K
asert-sinkhole.net	53	DNS	High	Low	7.7M	5.0K	400
nobelgroup.biz	25	SMTP	High	High	7.5M	580	107
teraz.biz	53	DNS	High	Medium	6.9M	600	45

also present security risks in the context of malware [5]. Similarly, Gruss et al. generalized the phenomenon of "use-after-free" exploits to the context of email addresses in which adversaries can not only re-activate specific, expired addresses from a free-mail provider, but also revive all the email addresses previously associated with an expired domain [65]. Finally, some works have investigated the possible abuse of expired domains in the context of remote JavaScript inclusions [7], Content Security Policies [8], and Android apps [66].

In contrast to the *bottom-up* approach of prior work, in this paper we used a *top-down*, target-agnostic approach of identifying expired domain names of potential value that escaped the attention of aggressive dropcatchers. The contributions of this approach are twofold: 1) confirming that the attack vector, which was originally discovered in a bottom-up fashion, is indeed feasible for an adversary to execute even if they were not targeting a specific entity, and 2) discovering the scope and severity of the potential harm that could be caused by an opportunistic attacker through the abuse of residual trust. Our results demonstrate that opportunistic adversaries may be granted access to millions of unique IP addresses situated in tens of thousands of autonomous systems merely by spending an average of $7.29 for the registration of each expired domain. Similar to our top-down approach, Borgolte et al. investigated the opportunity to hijack domain names in a target-agnostic fashion, by identifying dangling DNS resolutions on public clouds [67].

Similar in nature to how we extract historical DNS information to use as a feature for another process, there have been numerous methods that propose using some aspects of DNS information to detect malicious behavior. Some attempt to classify malicious websites with direct use of DNS information [68], [69]. Others propose proxy metrics that score malicious intent behind co-occurrence in DNS queries to blocklisted domains [70] and repeated queries to DNSBLs [71]. Lastly, researchers have proposed systems that attempt to predict future malicious behavior with only static properties such as nameserver properties and domain registration information [72], [73], [74].

VII. DISCUSSION & CONCLUSIONS

In this paper, we demonstrate the alarming potential abuse of residual trust with a systematic yet opportunistic approach that begins with re-registering previously popular expired domains and detecting residual trust traffic to these domains. In contrast to a bottom-up, targeted approach that involves the analysis of a specific system to discover its valuable domain names and the subsequent monitoring infrastructure required to detect when one of them may be dropped, our approach is simple yet powerful. It is target-agnostic, straightforward to implement, can be repeated indefinitely, and thus scales well with the number of domains. It also confirms that the worries posed by prior work that utilized a bottom-up approach are indeed realistic. Our approach re-registered domains that previously served as a torrent tracker, an API for a computer lab usage statistics service used by many universities, an API that was a point of contact for a common Android haptics library, security company DNS sinkhole servers, an Internet radio and music station, command-and-control centers for malware and potentially unwanted programs (PUPs), and an email tracker.

Ethics

We have taken a number of precautions in our experiment to ensure that we minimize the risk of harm to users and systems. We never advertised the domains that we registered thereby excluding one of the major ways that users discover websites (e.g., encounter them in a blog post or in social media). Moreover, we did not serve any content on them, except for the custom 404 HTTP(S) page that delivers our bot traps and JavaScript fingerprinting code. Although we stored raw packet captures, we never engaged with clients to further advance the communication. Thus, we stored only what the connecting clients initially sent us, and then indicated that the requested content was not found or terminated the connection. While it is in principle possible that our servers received actual user credentials, it is highly improbable. Our analysis for SSH, FTP, and Telnet traffic resulted in only one IP address tagged as `trust`, and it was a false positive as discussed in Section IV-C. It is also unlikely that we have collected any user credentials in our HTTP(S) traffic because our 404 page did not contain any forms that users could accidentally fill, and we did not serve any 401 Unauthenticated responses that would have challenged a client to send HTTP Basic Auth credentials. Any cached HTTP responses from the previous owners of the domains will have likely expired in the over 30-day period between the domain's expiration and our re-registration (Section II). By design, we know nothing about the connecting clients, hence the data that we collected is only valuable for the aggregate statistics that we presented in the paper.

We have slightly more information about the HTTP(S) clients that supported JavaScript from the modified FingerprintJS script that was served to connecting web clients along with the HTML response. FingerprintJS is a state-of-the-art fingerprinting library that extracts more than 30 types of attributes from a user's browser including information about the detected screen, the number of CPU cores, renderings of complicated canvas objects, and installed fonts. Our

script collected all the data that is made available by FingerprintJS, in addition to the client's geolocation, and stored them in their raw form, so that we could later perform the analyses and clustering that was required for the results presented throughout this paper. We only use the collected fingerprints as an indication that a real browser visited our website with a fully functioning JS engine. Therefore, in retrospect, we could have collected fewer attributes from the FingerprintJS library. This observation matches that of the recent work of Li et al. who observed that the vast majority of bots do not support JavaScript [33], making JavaScript support a good method to straightforwardly filter out unsophisticated bots.

We have no ability (or desire) to track users across the web (i.e., across different websites), hence our collected fingerprints *cannot* be used to harm user privacy through the recreation of their browsing sessions. Moreover, as Gómez-Boix et al. have shown, the browser fingerprints extracted from real-life users on popular websites, are significantly less unique than those collected from volunteers participating in privacy studies [75]. This reduced fingerprint uniqueness coupled with the drift of fingerprints (due to browser updates, new OS-level drivers for graphics cards, etc.), further reduces the potential of our now two-year-old fingerprinting data to harm a user's privacy.

In terms of contacting affected domains, by the time we were finished with our traffic analysis, all our registered domains had already expired and were returned to the pool of domains available for re-registration. Moreover, it is worth noting that the vast majority of the investigated domains were previously operated either by malicious actors or by companies that no longer exist, hence we have no information of whom to contact, if anyone, about the domains that were no longer under our control.

Limitations

Despite the advantages, there are inherent limitations in our top-down approach. For one, our method will not find valuable domains that do not have a large number of DNS resolutions (in our passive DNS database). A simple example of such domains may be domains with DNS records that use long time-to-live expiration values. Although a top-down approach may be able to find more valuable domains than individual bottom-up approaches, it may cost more to actually exploit the domains using a top-down approach. An adversary who uses a bottom-up, targeted approach will already have a deep understanding of the system and infrastructure of their target and, consequently, how they can be exploited. With a top-down approach, an adversary is likely to obtain domain names associated with a diverse array of services and applications. To exploit these domains, the adversary will need to analyze and implement infrastructure specific to each domain. Although we have shown that it is likely for a third party to successfully identify the type of service previously offered on an expired domain, it may be nontrivial to reverse engineer the details necessary for its exploitation.

False Positives

Given the highly-specific nature of our trust filters, particularly for HTTP(S) traffic, false positives are unlikely. We find domains with significant trust traffic when filtering for requests to *residual paths* (paths that used to exist on the domain on HTTP(S)). Residual paths require knowledge of the service offered by the domain before it expired. We consider all the possibilities for false positives from using residual paths as unlikely for the following reasons. First, indexers for search engines or archives are likely aware of residual paths, but are easily identifiable from their User-Agent header and are filtered out as bot traffic. Second, those who are unfamiliar with the previous service offered on the domain are unlikely to specifically request (i.e., guess) a residual path given that they only include service-relevant paths and exclude paths common to many domains. Also, because our containers serve the same content to all requests, no additional information is gained from multiple attempts. Last, we consider those who are unfamiliar with the previous service, but gained knowledge of it from third-party references (e.g., forum posts or search engine results). We argue that this is a form of residual trust because they, in effect, discovered a new service from third-party references and proceeded to visit the domain.

Another filter that could introduce false positives is our use of fingerprints, but we know from recent work that most bots and crawlers do not support JavaScript [33]; of those that do, they do not have full support or are still likely to fall for our bot traps.

A less obvious point that may have introduced more traffic and false positives does not stem from an analysis technique, but rather from obtaining a TLS certificate for each domain. Certificate Transparency (CT) [76] is a key component of the public key infrastructure ecosystem on the web that introduces publicly auditable certificate transparency logs. Although we did not explicitly advertise our domains anywhere, we indirectly advertised that our domains were issued TLS certificates and may, as a result, have received traffic from CT log monitors. However, we argue that these clients are still bots and therefore will most likely exhibit the same types of behavior that our bot-detection filters target. It is therefore unlikely that requests from CT monitors were flagged as residual trust-related traffic because of the nature of our HTTP(S) trust filters. Instead, it is more likely for these clients to have been flagged as `bot` or `neutral`.

Going forward

Our residual trust traffic detection pipeline demonstrated a clear and pressing potential abuse of residual trust in domains. Although there were some configurable variables in our experiment and analysis methodology that could be changed, such as the domain selection strategy and the specific passive DNS database, it is unlikely to result in contradictory findings. Attackers abusing residual trust can conduct a number of different attacks, ranging from stealing PII, infecting devices, reviving botnets, and even bypassing web security mechanisms. Even though there exist countermeasures for some specific instances of residual-trust abuse (such as the use of Subresource Integrity, or SRI, that can protect remote first-party websites against malicious changes in remote scripts [77]), there currently exist no recommended defense mechanisms against the overall problem of residual trust abuse.

An ideal defense mechanism would need to systematically define the integrity of a domain, accounting for features such as ownership and domain content, and determine whether it has changed or was compromised, such as the system proposed by Lever et al. [1]. In this context, we observe that the relatively recent move to anonymize WHOIS data may have increased the privacy of domain owners, but it also took away information that could be used to detect when a domain changed ownership.

Orthogonally to these types of change-detection systems, we argue that increased monitoring and asset-management capabilities have the potential to provide early warnings to developers and administrators. Given the domain-expiration timeline defined by ICANN (Section II), a domain will stop resolving for at least 30 days before it becomes publicly available for re-registration. With the right monitoring

infrastructure, this should enable everyone who depends on a domain to discover the impending drop and make the necessary changes *before* the domain changes hands.

In conclusion, in this paper we demonstrated that a top-down, target-agnostic approach that relies on past resolution data to identify residual-trust domains has the potential to offer attackers access to millions of users and thousands of systems, for the cost of a few, well-chosen, domain registrations. To characterize the potential of this attack, we re-registered 201 domains, developed traffic analysis pipelines to detect residual trust traffic, and successfully profiled 128 domains to develop domain-specific indicators of residual trust derived from the type of services they previously offered. We reported on the types of traffic we received and detailed the process of differentiating traffic received from bots and traffic due to residual trust via a number of filters based on architectural elements, request characteristics, and external IP blocklists. Even after our aggressive filtering, we showed how our pool of domain names attracted requests from over 2.6M IP addresses, which attempted to connect to torrent trackers, malware C&C servers, email trackers, college-lab software, and online radio stations. We hope that this paper rekindles the community's interest in detecting and protecting against cases of domain-name-based residual trust.

Acknowledgements: We thank our shepherd Kevin Borgolte and the anonymous reviewers for their helpful feedback as well as Farsight for giving us access to their passive DNS APIs. This work was supported by the Office of Naval Research (ONR) under grant N00014-20-1-2720, as well as by the National Science Foundation (NSF) under grants CMMI-1842020, CNS-1813974, and CNS-1941617.

References

[1] C. Lever, R. Walls, Y. Nadji, D. Dagon, P. McDaniel, and M. Antonakakis, "Domain-z: 28 registrations later measuring the exploitation of residual trust in domains," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 691–706.

[2] T. Lauinger, A. Chaabane, A. S. Buyukkayhan, K. Onarlioglu, and W. Robertson, "Game of registrars: An empirical analysis of post-expiration domain name takeovers," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 865–880.

[3] T. Lauinger, A. S. Buyukkayhan, A. Chaabane, W. Robertson, and E. Kirda, "From deletion to re-registration in zero seconds: Domain registrar behaviour during the drop," in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 322–328.

[4] N. Miramirkhani, T. Barron, M. Ferdman, and N. Nikiforakis, "Panning for gold. com: Understanding the dynamics of domain dropcatching," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 257–266.

[5] C. Lever, P. Kotzias, D. Balzarotti, J. Caballero, and M. Antonakakis, "A lustrum of malware network communication: Evolution and insights," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 788–804.

[6] E. Alowaisheq, P. Wang, S. Alrwais, X. Liao, X. Wang, T. Alowaisheq, X. Mi, S. Tang, and B. Liu, "Cracking the wall of confinement: Understanding and analyzing malicious domain."

[7] N. Nikiforakis, L. Invernizzi, A. Kapravelos, S. Van Acker, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "You are what you include: large-scale evaluation of remote javascript inclusions," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 736–747.

[8] S. Roth, T. Barron, S. Calzavara, N. Nikiforakis, and B. Stock, "Complex security policy? a longitudinal analysis of deployed content security policies," in *Proceedings of the 27th Network and Distributed System Security Symposium (NDSS)*, 2020.

[9] "Domain name industry brief (dnib). https://www.verisign.com/en_us/domain-names/dnib/index.xhtml," https://www.verisign.com/en_US/domain-names/dnib/index.xhtml.

[10] H. Hu, P. Peng, and G. Wang, "Characterizing pixel tracking through the lens of disposable email services," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 365–379.

[11] "Expired registration recovery policy. https://www.icann.org/resources/pages/errp-2013-02-28-en," https://www.icann.org/resources/pages/errp-2013-02-28-en.

[12] C. Evans, C. Palmer, and R. Sleevi, "Public key pinning extension for http," Internet Requests for Comments, RFC Editor, RFC 7469, 04 2015. [Online]. Available: https://tools.ietf.org/html/rfc7469

[13] M. Contributors, "Http public key pinning (hpkp)," https://developer.mozilla.org/en-US/docs/Web/HTTP/Public_Key_Pinning.

[14] C. Palmer, "Intent to deprecate and remove: Public key pinning," https://groups.google.com/a/chromium.org/g/blink-dev/c/he9tr7p3rZ8/m/eNMwKPmUBAAJ?pli=1.

[15] F. Weimer, "Passive dns replication," in *FIRST conference on computer security incident*, vol. 98, 2005.

[16] "Passive dns historical internet database: Farsight dnsdb," https://www.farsightsecurity.com/solutions/dnsdb/.

[17] P. Agten, W. Joosen, F. Piessens, and N. Nikiforakis, "Seven months' worth of mistakes: A longitudinal study of typosquatting abuse," in *Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS 2015)*. Internet Society, 2015.

[18] P. Kintis, N. Miramirkhani, C. Lever, Y. Chen, R. Romero-Gómez, N. Pitropakis, N. Nikiforakis, and M. Antonakakis, "Hiding in plain sight: A longitudinal study of combosquatting abuse," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 569–586.

[19] N. Nikiforakis, M. Balduzzi, L. Desmet, F. Piessens, and W. Joosen, "Soundsquatting: Uncovering the use of homophones in domain squatting," in *International Conference on Information Security*. Springer, 2014, pp. 291–308.

[20] N. Nikiforakis, S. Van Acker, W. Meert, L. Desmet, F. Piessens, and W. Joosen, "Bitsquatting: Exploiting bit-flips for fun, or profit?" in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 989–998.

[21] Y.-M. Wang, D. Beck, J. Wang, C. Verbowski, and B. Daniels, "Strider typo-patrol: Discovery and analysis of systematic typo-squatting." *SRUTI*, vol. 6, no. 31-36, pp. 2–2, 2006.

[22] "Domainmonster," https://www.domainmonster.com/contact/.

[23] "Dynadot," https://www.dynadot.com/.

[24] "Namejet," https://www.namejet.com/default.aspx.

[25] "Pool," https://www.pool.com/.

[26] "Snapnames," https://www.snapnames.com.

[27] "Let's encrypt," https://letsencrypt.org/.

[28] "Cowrie, a medium to high interaction ssh honeypot. https://github.com/lanjelot/twisted-honeypots," https://github.com/lanjelot/twisted-honeypots.

[29] "Twisted honeypots. https://github.com/lanjelot/twisted-honeypots," https://github.com/lanjelot/twisted-honeypots.

[30] "Elasticsearch. https://www.elastic.co/," https://www.elastic.co/.

[31] "Fingerprintjs." https://github.com/fingerprintjs/fingerprintjs.

[32] K. Park, V. S. Pai, K.-W. Lee, and S. B. Calo, "Securing web service by automatic robot detection." in *USENIX Annual Technical Conference, General Track*, 2006, pp. 255–260.

[33] X. Li, B. A. Azad, A. Rahmati, and N. Nikiforakis, "Good bot, bad bot: Characterizing automated browsing activity," in *IEEE Symposium on Security and Privacy*, 2021.

[34] "Browser detection using the user agent," https://developer.mozilla.org/en-US/docs/Web/HTTP/Browser_detection_using_the_user_agent#making_the_best_of_user_agent_sniffing.

[35] "Path traversal," https://owasp.org/www-community/attacks/Path_Traversal.

[36] "Blindelephant web application fingerprinter," http://blindelephant.sourceforge.net/.

[37] "Diversified sampler aggregation: Elasticsearch reference [7.12]," https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-bucket-diversified-sampler-aggregation.html.

[38] "Significant text aggregation: Elasticsearch reference [master]," https://www.elastic.co/guide/en/elasticsearch/reference/master/search-aggregations-bucket-significanttext-aggregation.html.

[39] "Kali tools wordlists package," https://tools.kali.org/password-attacks/wordlists, Feb 2014.

[40] T. Mursch, "Engenius routers found in mirai-like botnet," https://badpackets.net/engenius-routers-found-in-mirai-like-botnet/, Mar 2019.

[41] D. Piscitello and G. Aaron, "Icann blogs," Nov 2017. [Online]. Available: https://www.icann.org/en/blogs/details/reputation-block-lists-protecting-users-everywhere-1-11-2017-en

[42] C. Tsaousis, http://iplists.firehol.org/.

[43] "Spamhaus," https://www.spamhaus.org/faq/section/DNSBLUsage.

[44] "Spam and open-relay blocking system (sorbs)," http://www.sorbs.net/.

[45] "Barracuda reputation block list (brbl)," https://www.barracudacentral.org/rbl.

[46] "Nicecast 1.10.1 manual," https://my.rockhost.com/dl.php?type=d&id=10.

[47] "Music player daemon 0.22.7 user's manual," https://www.musicpd.org/doc/html/user.html.

[48] "Labstats," https://labstats.com.

[49] S. Englehardt, J. Han, and A. Narayanan, "I never signed up for this! privacy implications of email tracking," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 1, pp. 109–126, 2018.

[50] "Android apk: com.movinapp.dict.ensv.free.apk 2.1.1 (english swedish dict.)," https://aapks.com/apk/english-swedish-dictionary-free/.

[51] "Android apk: com.movinapp.dict.essv.free.apk 3.9.6 (offline spanish swedish dictionary)," https://aapks.com/apk/spanish-swedish-dict-free/.

[52] "Android apk: com.stevenschoen.emojiswitcher (emoji switcher)," https://emoji-switcher.en.uptodown.com/android.

[53] "Avantmobile facebook profile," https://www.facebook.com/avantmobile/.

[54] "Avantmobile instagram profile," https://www.instagram.com/avantmobile/.

[55] "Maltivese scan of facecommute.com," https://maltiverse.com/hostname/facecommute.com.

[56] "Virustotal graph: gb-installer.gbox-data.net," https://www.virustotal.com/graph/gb-installer.gbox-data.net.

[57] R. Bolstridge October 31, "Dyn ddos attack: Wide-spread impact across the financial services industry (part 1)," https://blogs.akamai.com/2016/10/dyn-ddos-attack-wide-spread-impact-across-the-financial-services-industry-part-1.html, Oct 2016.

[58] C. Matlack, "French websites knocked offline in cyber-attack on cedexis," https://www.bloomberg.com/news/articles/2017-05-10/french-websites-knocked-offline-in-cyber-attack-on-cedexis, May 2017.

[59] "Arbor networks," https://www.netscout.com/company.

[60] "Wayback machine: smartereum.com (11/10/2019)," http://web.archive.org/web/20191110090712/https://smartereum.com/.

[61] "Wayback machine: forklog.media (02/20/2020)," http://web.archive.org/web/20200220011449/https://forklog.media/.

[62] "Wayback machine: profitgid.ru (08/22/2019)," http://web.archive.org/web/20190822204341/profitgid.ru.

[63] "Telegram: Leonbets," https://xn--r1a.website/s/bk_leonbets?before=82.

[64] T. Moore and R. Clayton, "The ghosts of banking past: Empirical analysis of closed bank websites," in *International Conference on Financial Cryptography and Data Security*. Springer, 2014, pp. 33–48.

[65] D. Gruss, M. Schwarz, M. Wübbeling, S. Guggi, T. Malderle, S. More, and M. Lipp, "Use-after-freemail: Generalizing the use-after-free problem and applying it to email services," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 2018, pp. 297–311.

[66] E. Pariwono, D. Chiba, M. Akiyama, and T. Mori, "Don't throw me away: Threats caused by the abandoned internet resources used by android apps," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 2018, pp. 147–158.

[67] K. Borgolte, T. Fiebig, S. Hao, C. Kruegel, and G. Vigna, "Cloud Strife: Mitigating the Security Risks of Domain-Validated Certificates," in *Proceedings of the 25th Network and Distributed System Security Symposium (NDSS)*.

[68] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious urls," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 1245–1254.

[69] D. Canali, M. Cova, G. Vigna, and C. Kruegel, "Prophiler: a fast filter for the large-scale detection of malicious web pages," in *Proceedings of the 20th international conference on World wide web*, 2011, pp. 197–206.

[70] K. Sato, K. Ishibashi, T. Toyono, H. Hasegawa, and H. Yoshino, "Extending black domain name list by using co-occurrence relation between dns queries," *IEICE transactions on communications*, vol. 95, no. 3, pp. 794–802, 2012.

[71] A. Ramachandran, N. Feamster, and D. Dagon, "Revealing botnet membership using {DNSBL} counter-intelligence," in *2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet ({SRUTI} 06)*, 2006.

[72] M. Felegyhazi, C. Kreibich, and V. Paxson, "On the potential of proactive domain blacklisting." *LEET*, vol. 10, pp. 6–6, 2010.

[73] S. Hao, M. Thomas, V. Paxson, N. Feamster, C. Kreibich, C. Grier, and S. Hollenbeck, "Understanding the domain registration behavior of spammers," in *Proceedings of the 2013 conference on Internet measurement conference*, 2013, pp. 63–76.

[74] J. Spooren, T. Vissers, P. Janssen, W. Joosen, and L. Desmet, "Premadoma: An operational solution for dns registries to prevent malicious domain registrations," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 557–567.

[75] A. Gómez-Boix, P. Laperdrix, and B. Baudry, "Hiding in the crowd: an analysis of the effectiveness of browser fingerprinting at large scale," in *Proceedings of the 2018 World Wide web Wonference*, 2018, pp. 309–318.

[76] B. Laurie, A. Langley, and E. Kasper, "Certificate transparency," Internet Requests for Comments, RFC Editor, RFC 6962, 06 2013. [Online]. Available: https://tools.ietf.org/html/rfc6962

[77] "Subresource integrity," https://www.w3.org/TR/SRI/, Jun 2016.