Uninvited Guests: Analyzing the Identity and Behavior of Certificate Transparency Bots

Brian Kondracki
Stony Brook University
bkondracki@cs.stonybrook.edu

Johnny So
Stony Brook University
josso@cs.stonybrook.edu

Nick Nikiforakis
Stony Brook University
nick@cs.stonybrook.edu

Abstract

Since its creation, Certificate Transparency (CT) has served as a vital component of the secure web. However, with the increase in TLS adoption, CT has essentially become a defacto log for all newly-created websites, announcing to the public the existence of web endpoints, including those that could have otherwise remained hidden. As a result, web bots can use CT to probe websites in real time, as they are created. Little is known about these bots, their behaviors, and their intentions.

In this paper we present CTPOT, a distributed honeypot system which creates new TLS certificates for the purpose of advertising previously non-existent domains, and records the activity generated towards them from a number of network vantage points. Using CTPOT, we create 4,657 TLS certificates over a period of ten weeks, attracting 1.5 million web requests from 31,898 unique IP addresses. We find that CT bots occupy a distinct subset of the overall web bot population, with less than 2% overlap between IP addresses of CT bots and traditional host-scanning web bots. By creating certificates with varying content types, we are able to further sub-divide the CT bot population into subsets of varying intentions, revealing a stark contrast in malicious behavior among these groups. Finally, we correlate observed bot IP addresses into campaigns using the file paths requested by each bot, and find 105 malicious campaigns targeting the domains we advertise. Our findings shed light onto the CT bot ecosystem, revealing that it is not only distinct to that of traditional IP-based bots, but is composed of numerous entities with varying targets and behaviors.

1 Introduction

Security of the modern web is reliant on the HTTPS protocol and, thus, the certificate authorities (CAs) that lay the groundwork of trust through the issuance of TLS certificates. This trust was shaken in 2011 however, when the popular CA DigiNotar misissued TLS certificates for Google domains, allowing for the exploitation of numerous Iranian users with man-in-the-middle attacks [14]. This was just one of many similar incidents affecting CAs in the early 2010s [1, 12, 36].

In response to these events, the *Certificate Transparency* [9] (CT) system was introduced to provide clarity and insight into the actions of CAs. CT works by logging the registration of all TLS certificates to public append-only logs. This allows domain owners to search for illegitimate registrations of certificates for their domains, and the public to audit the actions of CAs. In 2015, Google Chrome began requiring all new Extended Validation certificates to be publicly-logged in the CT system, with this extending to all new certificates in 2018 [11]. Today, this participation is also required by popular browsers such as Apple's Safari [4] and Opera Browser [34]. However, while Mozilla has pledged support for CT development and use [29], it is currently not enforced by Firefox browser [16].

Because of the growing use of TLS on the web [21], and the pressure applied by browser vendors for all sites to serve content over HTTPS, CT has essentially become a log for all newly created websites. This includes not only new benign websites (or existing websites who are renewing their expiring certificates) but also malicious websites which also need to use TLS (and therefore CT) to properly render without warnings in a user's browser.

Prior work has capitalized on this constant stream of websites to, among others, identify phishing websites the moment they go online [46, 48, 54, 67], as well as provide links for further indexing of the web [19, 38]. As these studies demonstrate, the massive volume of certificate registrations makes it impractical for analysts to manually visit each newly created website. Thus, this job is often delegated to automated web bots that tail CT logs, either visiting every domain or seeking out specific domains of interest. As of yet, however, this web bot ecosystem has not been fully studied to determine the sources and behaviors of these bots, including its use by attackers to discover new targets as they are created.

In this work, we create a honeypot infrastructure we call CTPOT, that allows us to continually create TLS certificates for pseudo-randomly generated subdomains and measure the network traffic generated by their inclusion on CT logs, from a number of network vantage points. Our certificate creation strategy centers around three distinct domain-content

categories, allowing us to observe the varying behaviors of bots with different goals.

Using CTPOT, we create 4,657 TLS certificates across 12 measurement nodes for pseudo-random subdomains corresponding to: popular trademarks, common web endpoints of web application software, and dictionary words which act as a baseline through which we compare and interpret the results of our measurement groups. Over the course of ten weeks, CTPOT received a total of 1.5 million web requests from 31,898 unique IP addresses, as well as 22,839 requests to SSH, FTP, and Telnet honeypots. As each domain generated by CTPOT was previously unused and completely unguessable, our curated dataset consists *entirely* of bot traffic.

By analyzing our curated dataset, we observe distinct behaviors from the bots targeting different types of subdomains, allowing us to learn about the intentions of these various bot populations. For instance, we observe bots targeting subdomains of common web application software send more than one request over 40% more often than bots targeting domains impersonating popular trademarks. Moreover, these bots exhibit more malicious behavior with over twice as many unique IP addresses attempting to authenticate with exposed network services such as SSH. Additionally, we correlate requests from seemingly isolated IP addresses into campaigns of related bots. Alarmingly, through this analysis we find 105 malicious campaigns attempting to perform malicious actions such as data exfiltration, fingerprinting, and vulnerability exploitation.

Our main contributions are as follows:

- We design and implement CTPOT, a honeypot-based system to create TLS certificates for pseudo-random subdomains, and analyze requests directed towards them. Using this system, we create 4,657 TLS certificates.
- We curate the first public dataset of CT bots. Analysis of this dataset yields valuable insight into their populations and behaviors, including the varying behaviors of bots with distinct objectives and targets.
- We correlate the behaviors of seemingly isolated bots into campaigns, finding 105 clusters of requests that are malicious in nature.

2 Background

In this section, we provide the required background information on CT and automated browsing to assist reader understanding for the remainder of the paper.

Certificate Transparency

The purpose of CT is to prevent the exploitation of users by ensuring certificates are not issued to malicious actors. This is achieved by logging all certificate registrations on publicly-available, append-only logs; allowing domain owners to monitor for invalid certificate registrations for their domain, and the community to audit the actions of CAs.

Though commonly managed by large organizations, anyone can create a CT log and advertise its contents to browsers. Likewise, anyone can submit a certificate to a CT log, though this is typically done by CAs during the process of certificate creation. When a certificate is submitted to a log, the log responds with a Signed Certificate Timestamp (SCT), which is a promise that the certificate will be added to the log either immediately or in the near future. The CA then attaches the SCT to the newly-issued certificate, which is proof the certificate was publicly logged in the CT system. Popular browsers such as Google Chrome and Safari enforce CT compliance by displaying errors when users visit HTTPS websites that present certificates lacking valid SCTs. This helps coerce CA participation, leading to over 90% of CAs logging to CT [10].

By simply hooking into readily available APIs [7, 15, 18], anyone can audit CT logs for signs of inappropriately created certificates (indicating a server compromise or a compromised/misbehaving CA). However, because of growing CT compliance by CAs, along with the proliferation of HTTPS usage across the web, these APIs indirectly provide visibility into a large percentage of all newly-created websites as they come online. Moreover, as the primary focus of CT is the public advertisement of new domains, observers of these logs can receive powerful information, including the existence of otherwise hidden web endpoints. This includes Fully Qualified Domain Names (FQDNs) that would not have been discovered via crawling or guessing (e.g. `test-deployment-888.example.com`) as well as top-level domain names (TLDs) from registries that do not share their Zone Records with the public (such as in Country-Code TLDs). Prior work by Scheitle et. al. demonstrated that web bots monitor CT logs, and initiate requests towards the FQDNs included on them [68].

Bots and Automated Browsing

The immense scale of the Internet makes it impractical for one looking to understand its trends and intricacies to do so by manual means. Rather, this job is delegated to automated browsing tools, commonly known as "spiders", "crawlers", or "bots". Web bots are often used for benign tasks such as search engine indexing [19, 38], and phishing detection [33, 75]. However, web bots are also responsible for many of the attacks on systems across the Internet [58].

Identification of bots has become a priority for website administrators, leading to to development and use of anti-bot services [40]. Prior work has shown effective bot detection techniques including fingerprinting of each client's characteristics and behavior [51, 61, 71]. The sophistication of web bots is often dependent upon the expected usage of these anti-bot fingerprinting techniques. Simple request tools such as wget [37] allow for quick scanning of a large number of websites using raw HTTP requests, but lack the traits of a real browser, increasing the probability of fingerprinting and subsequent blocking. Conversely, utilization of fully-instrumented browsers using libraries such as Selenium [35] allow web bots to more-closely resemble real users, at the expense of scanning efficiency.

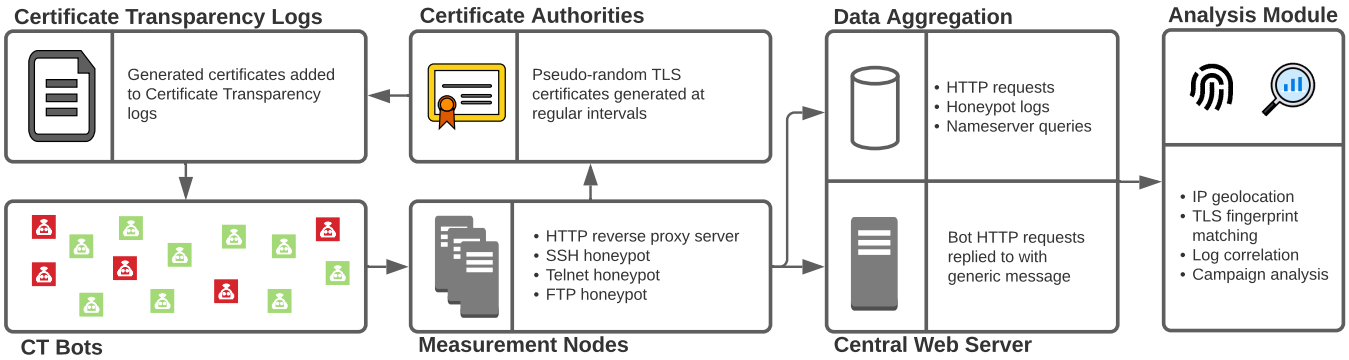

Figure 1: *Architecture of* CTPOT*, our CT honeypot system composed of a series of measurement nodes that periodically generate certificates for pseudo-random domains and record the traffic generated towards various network services.*

3 Methodology & Experimental Setup

In this section, we first describe the design of CTPOT, our system used to create TLS certificates and collect data on bots monitoring CT logs. Additionally, we describe the details of deploying CTPOT in the wild.

3.1 CTPOT System Design

To study the characteristics and behaviors of bots consuming CT logs, we design a distributed honeypot system that records data from a number of network vantage points. This system, which we call CTPOT, is illustrated in Figure 1. It consists of two main modules: a series of decentralized measurement nodes, and a central data-aggregation node. This "hub-and-spoke" architecture emphasizes scalability as measurement nodes can be added or removed from the system without interrupting the others. Meanwhile, recorded data is periodically pulled by the data-aggregation node for analysis.

3.1.1 Measurement Nodes

Each CTPOT measurement node hosts an Apache TrafficServer [2] reverse proxy server, listening for both HTTP and HTTPS requests. All valid HTTP(S) requests are forwarded to the web server located on the data-aggregation node, without any caching enabled. As mentioned earlier, this reverse-proxy design allows us to freely add and remove nodes to the infrastructure without the need to update any centralized state. Additionally, spreading our experimental footprint across multiple IP addresses decreases the chances of bots blocklisting our nodes.

Since it is straightforward for bots to misidentify themselves in their User-Agent headers (e.g. a `wget` client presenting the User-Agent of a popular web browser), we attempt to identify each visitor connecting over HTTPS with a TLS fingerprint. To do this, we enable the JA3 TLS fingerprinting plugin for Apache TrafficServer [3]. JA3 fingerprints are created by concatenating the following fields from each TLS Client Hello message: protocol version, accepted ciphers, supported extensions, elliptic curves, and elliptic curve formats. The MD5 hash of the resulting string is the unique identifier for that client [32]. Prior work has demonstrated that the list of supported cipher suites in a TLS Client Hello message is enough to reliably identify the underlying client platform in HTTPS traffic [50]. To communicate the JA3 fingerprint and other information about the requesting client (e.g., client IP address and request scheme), Apache TrafficServer appends a series of HTTP headers to the original request with the additional information before it is forwarded to the data-aggregation node.

In addition to probing the web servers of publicly-accessible machines, malicious bots may also scan additional ports in order to discover vulnerable services which can be abused. To determine the extent to which CT bots interact with these network services, we include an SSH and Telnet honeypot [13], as well as an FTP honeypot [20] on each measurement node. We use each of these three honeypots in a "low-interaction" fashion, meaning visitors are presented with a login prompt, but can never actually authenticate. Rather, they are presented with a message indicating their authentication attempt failed. For the purpose of this study, any visitor attempting to authenticate with these honeypots can be considered malicious, regardless of the actions they would take upon successful authentication.

Certificate Registration

Unlike bots that utilize IP-based scans to discover publicly-accessible services across large Internet subnets, CT bots have the ability to filter targets based on the content of their domain names. For instance, a malicious bot that is targeting outdated WordPress websites could opt to only make requests towards domains that contain strings related to blogs and WordPress (e.g., `wordpress.example.com` and `blog.example.com`). Therefore, we carefully design a certificate registration strategy that allows us to separate the overall CT bot population into groups of varying intentions. Specifically, we seek to observe distinct populations of malicious and benign bots that feed off of CT logs.

To do so, we define three TLD+2 subdomain content groups, allowing us to not only separate the CT bots with these intentions, but also separate their behaviors from the bots which scan all domains that appear on CT logs. To attract bots operated by security companies and phishing researchers, we create subdomains that contain the names of companies known to be common targets of phishing attacks (e.g., Apple, Paypal, Facebook, etc.,) [5], along with a qualifier string (e.g., reviews, tutorials, lessons, etc.,). For example, we reason that the FQDN `paypal-reviews.example.net` is likely to trigger the automated crawling logic of a CT bot searching for domains with common phishing targets and then be dismissed during manual review when the operator reads it in the context of the rest of the string (i.e. `paypal-reviews`). We use these qualifying phrases, along with a message indicating the purpose of our study within all HTTP(S) responses (described in Section 3.1.2), in an attempt to prevent automatic blocking by anti-phishing entities, so that CTPOT can keep attracting both benign and malicious CT bots for the entirety of our experiment.

To attract malicious bots, we create subdomains that are targeted by popular vulnerability-scanning software [31] (e.g., wp-admin, sql, db, etc.,). Our rationale is that a certain class of resource-constrained attackers may only focus their attention on hosts that are likely to be running certain web applications, as opposed to wasting their resources on any and all online hosts.

Lastly, to compare the bot populations of the previous two groups against bots that indiscriminately scan domains on CT, we create subdomains containing the names of various fruits, vegetables, and colors. We refer to these three groups as: `Impersonating`, `Sensitive`, and `Baseline`, respectively. A full list of all strings used to construct the subdomains of these groups is located in Table 5 of the Appendix.

For the purposes of this study, we must ensure all visitors to the domains we advertise on CT discovered them through consumption of CT logs. Therefore, in addition to the TLD+2 subdomains which categorize our advertised domains, we add a pseudo-random TLD+3 subdomain which is highly unlikely to be guessed. Specifically, this subdomain encodes the timestamp the certificate was registered along with five random characters. By including this information, we can attribute all requests to these domains to CT bots, with high-confidence.

All certificates are created on each measurement node by randomly choosing one of the three described groups, and then a string from the chosen group. In the case of Impersonating domains, a randomly chosen trademark is appended to a randomly chosen qualifier string (i.e., *facebook-reviews*). Each generated subdomain is appended to the primary domain assigned to that particular measurement node. Table 1 shows an example of each domain category, along with the encoded certificate creation timestamp.

These certificates are created using the Let's Encrypt Certbot API [8]. The rate limits enforced by Let's Encrypt restrict a single IP address to 50 new certificates each week [25]. Therefore, each CTPOT measurement node creates one certificate every four hours, for a total of six per day or 42 per week. This provides ample daily coverage, while also allowing for some margin of error to ensure API limits are not exceeded. Our certificate creation strategy does not place any undue burden on Let's Encrypt servers, accounting for only 0.004% of all certificate requests each day [26].

Table 1: *Example domains of each measurement category, and their corresponding encoded certificate creation timestamps. For brevity, the primary domain is excluded from each example.*

Example Subdomain	Type	Encoded Timestamp
`jjr20201wvo180002.zoom-help`	Impersonating	2022-02-01T18:00:02
`bwr11215lkj013247.wp-admin`	Sensitive	2021-12-15T01:32:47
`yug11031wvo061216.blue`	Baseline	2021-10-31T06:12:16

3.1.2 Data Aggregation Node

To allow for flexibility in the number of measurement nodes used at any given time, CTPOT uses a single data-aggregation node to gather all data collected from each measurement node in one location for analysis and processing. The data-aggregation node hosts an Nginx [30] web server in which measurement nodes forward all requests to. These requests are stored in a database for future processing, and are responded to with a simple HTML webpage explaining the purpose of this experiment along with additional contact details. We chose to disclose the presence of our study to all visitors as an additional attempt to avoid getting flagged as malicious by the operators of anti-phishing tools.

Additionally, this node also hosts a Bind9 [6] authoritative nameserver responsible for all domains used in our study. This nameserver logs basic information about all queries received, as well as the eDNS subnet of the querying client (if included by the client's resolver), allowing us to determine the subnet of the querying stub resolver.

As previously mentioned, each measurement node hosts SSH, Telnet, and FTP honeypots. The log files produced by these honeypots are periodically collected by the data-aggregation node and stored for future analysis. These logs, along with the aforementioned HTTP and DNS data, are fed into the analysis module for processing.

3.2 Cheap vs. Expensive TLDs

It is entirely possible that bots utilizing CT to discover scanning targets are sensitive to the content of domain names. This includes both the primary domain, subdomains, as well as the chosen TLDs. For instance, studies have shown that a large fraction of phishing sites are hosted on cheap TLDs where attackers can keep registering new low-cost domains whenever their existing ones are blocked [39, 60]. As such, operators of anti-phishing tools may prioritize the scanning of low-cost domain names that appear in CT logs, over established (and

more expensive) domains. Contrastingly, malicious CT bots searching for victims may focus their scanning resources on domains registered on well-established TLDs, using registration price as a proxy variable for the value of the targeted host.

To determine whether different TLDs need to be part of our certificate-creation strategy, we conducted a small pilot experiment with CTPOT using two measurement nodes. Each node had the same primary domain with only the TLD changing: one node operated a low-cost *.xyz* domain whereas the other host operated a traditional *.com* domain name. Additionally, we fixed the random seed used to construct the subdomains for all certificate registrations such that both nodes created certificates for the same domains, at the same times, with only the TLD changing.

We ran this pilot experiment for one week, and monitored the requests we received from bots. In total, we received 1,508 requests towards the *.com* domain from 421 unique IP addresses. Conversely, our *.xyz* domain received 3,347 requests from 633 unique IP addresses. Additionally, over half of the IP addresses that requested content from the *.com* domain, also requested content from the *.xyz* domain. As a result, we decided to use *.xyz* TLDs for our main experiments allowing us to both attract more clients as well as lower the operating costs of our study. We discuss the potential limitations of utilizing a single TLD in Section 6.2.

3.3 Deployment and Data Collection

Using the described system setup, we created a total of 12 CTPOT measurement nodes. Six of these nodes were located in a popular public cloud, and the remaining six in our institution's datacenter.

To isolate all measurement nodes, each node is assigned a dedicated primary domain to which subdomains for that node are applied to. We designed a separate generation strategy for these primary domains to ensure that they do not interfere with the decision of a CT bot to interact with that subdomain. This strategy involves concatenating a randomly chosen string from each of three groups: names of trees, names of flowers, and names of birds–with all potential trademark conflicts removed.

Each measurement node creates a new certificate every four hours, on the top of the hour. We stagger this process on each measurement node such that there are at least three certificates being created every hour. This results in a total of 72 certificate creations each day. However, this number varies depending upon the behavior of the Let's Encrypt API (e.g., certificate creation failures would result in fewer created in a particular hour).

4 CT Bot Analysis

In this section, we report the findings of our deployment of CTPOT for ten weeks, from November 3, 2021 to January 12, 2022. In total, we created 4,657 TLS certificates, each for a unique domain. For the rest of this section, unless otherwise noted, we focus on the traffic that CTPOT received targeting one or more of our advertised FQDNs. This is to ensure that we do not attribute the general scanning activity (that publicly-reachable hosts receive from Internet-wide scanners) as related to the Certificate Transparency mechanism.

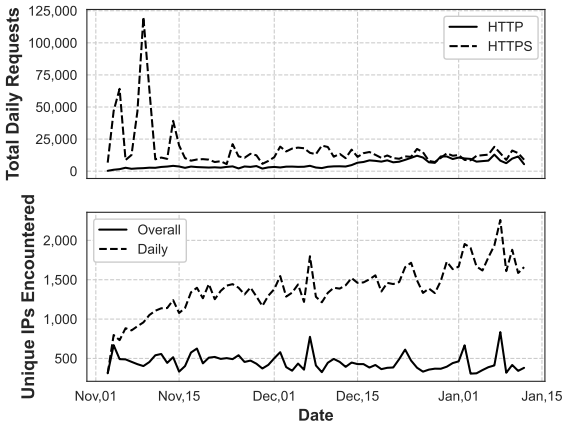

Figure 2: *(Top) Total daily requests to our measurement infrastructure over both HTTP and HTTPS. (Bottom) Total new unique IP addresses encountered each day of our measurement period both respective to the current day (dotted line), and throughout the entire data collection period (solid line).*

Over the course of our data collection period, CTPOT recorded 1.5 million HTTP requests from 31,898 unique IP addresses towards the domains advertised on CT. This corresponds to an average of 129,723 requests from 2,658 unique IP addresses, per measurement node. As the domains advertised on CT are random in nature and have not existed prior to the creation of each certificate, we can consider all such requests to have originated from web bots. Furthermore, as these domains were only ever advertised on CT, we can conclude that the bots that made these requests did so after observing one of our domains on a public CT log. To the best of our knowledge, this is the first comprehensive dataset on CT bots, detailing their identities and distinct behaviors. To assist the community in understanding the CT bot ecosystem, and defending against malicious actors therein, we are making our dataset publicly available to researchers (Section 8).

4.1 CT Bot Traffic

Figure 2 (Top) shows the total daily HTTP and HTTPS requests from visitors to our CTPOT deployment. We find that the number of requests remains fairly stable throughout our entire data-collection period, averaging around 10K-15K total daily requests on both HTTP and HTTPS. In early November 2021, CTPOT encountered a series of daily request spikes of over 100K. Upon close inspection of requests on these days, we find this to be the result of the actions of a single CT bot. More details regarding this bot can be found in Section 5.

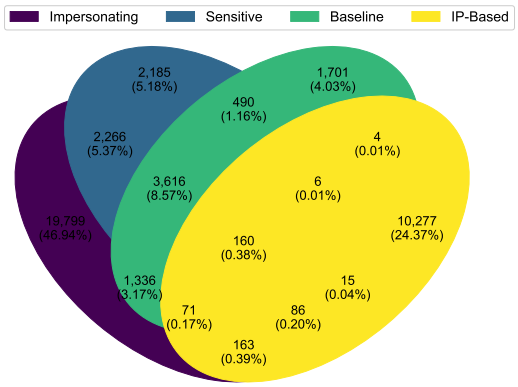

Figure 3: *Unique IP addresses targeting each of the three domain types in our study, along with the IP addresses of each measurement node. We note that each circle in this figure is not to scale, but rather serves to demonstrate the relationships between each group of CT bots in our dataset.*

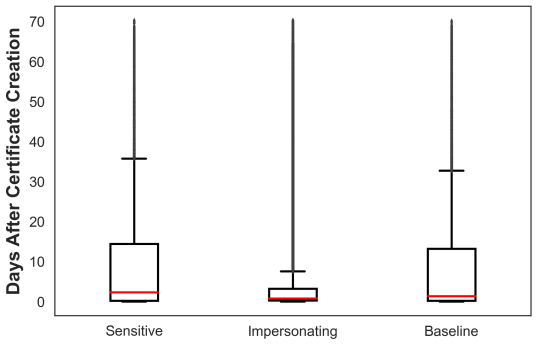

Figure 4: *Number of days after certificate creations our measurement nodes received HTTP(S) requests from CT bots.*

The overall scale of the CT bot ecosystem is clearly demonstrated in Figure 2 (Bottom). In this figure, the dotted line represents the total number of unique IP addresses encountered on each day, while the solid line represents only the unique IP addresses that first interacted with our infrastructure on that particular day (i.e., on December 1, our infrastructure saw roughly 500 new IP addresses that it had not seen in any of the days prior). We observe a steady stream of new visitors interacting with CTPOT each day, with approximately 300-500 previously unseen IP addresses recorded daily. Moreover, in the final days of our data collection period, we even see large spikes in this number, emphasizing the constant expansion of the CT bot IP address pool.

CT Bot Populations

Figure 3 shows the number of unique IP addresses that targeted each group of domains we advertised on CT, along with bots that targeted the IP address of each measurement node. We note that our measurement nodes also received traffic from an additional 4,855 IP addresses that targeted only the primary domains advertised on CT. However, as we cannot be certain of the subdomain, and thus source of the visitor, we discard these requests from our dataset.

Overall, we find that CT bots occupy a distinct subset of the overall web bot population, with less than 2% overlap between CT bots and IP-based bots (i.e., those that sent requests towards the IP address of the particular measurement node, rather than the domain listed on CT). Currently, IP-based bots do not utilize the additional information provided by CT to target their probes, opting instead to scan IP address subnets. Additionally, we find the overall size of the CT bot population to be over twice that of IP-based bots in our dataset.

Among CT bots, we find that the majority of IP addresses targeted only `Impersonating` domains, completely ignoring the two additional groups. This shows how there exists a large population of bots that are actively filtering CT logs for domains with specific content. In this case, these bots may be searching for newly-created phishing websites that are targeting popular trademarks. We contrast this with the IP addresses that visited `Sensitive` and `Baseline` domains, which exhibit a much larger overlap with each other as well as with `Impersonating` domains. This indicates that these bots are not filtering based on the content of domain names, opting rather to visit all — or a large percentage of all — domains listed on CT logs. We reason that bots with this behavior fall into one of two categories: academic or industry scanners indexing the web, or malicious bots looking for attack targets. Distinctions between these two groups can be made by observing their varying behaviors, which we do later in this section.

Interestingly, we also find a number of bots that only visited `Baseline` domains. That is, domains that simply contain non-sensitive dictionary words. We attribute these isolated requests to the non-deterministic nature of CT log streams where it is not guaranteed any particular observer of a log will encounter all certificate registrations. This non-determinism can be attributed to clients connecting and disconnecting to any given CT log, as well as not connecting to all possible logs operated by different CAs [56].

CT Bot Request Properties

Figure 4 shows the number of days after the creation of all certificates that our measurement nodes received all requests. On average, the first requests towards our domains occurred 5.9 minutes after a certificate was registered for that domain, consistent across all three domain groups. In some cases, we observed the first request arriving as early as 12 seconds after certificate creation.

Additionally, all three groups continued to see periodic requests long after certificate creation, indicating that a subset of CT bots are interested in long-term changes to domains that appear on CT logs. However, we find this is much more common in bots that target `Sensitive` and `Baseline` domains, rather than bots that target `Impersonating` domains. As bots in the latter group may be part of the infrastructures of anti-phishing entities, they are more likely to send one or a small number

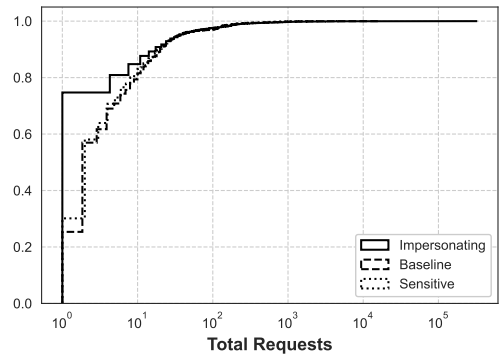

Figure 5: *Distributions of total requests from each unique IP address by subdomain type.*

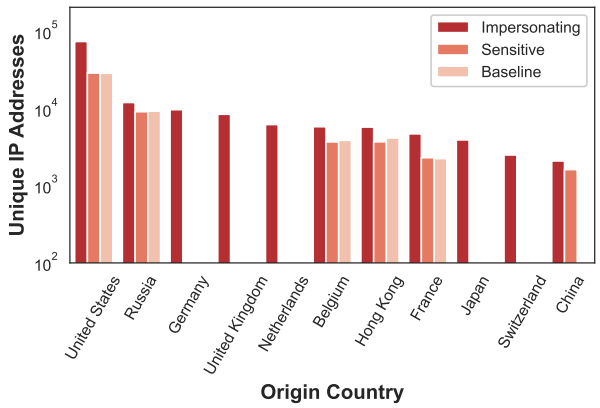

Figure 6: *Top geographic locations of IP addresses that requested content from our measurement infrastructure.*

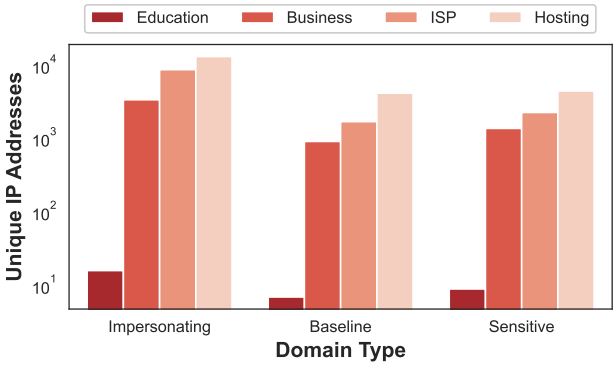

Figure 7: *ASN types of IP addresses that requested content from our measurement infrastructure.*

of requests shortly after encountering a domain to analyze for phishing traits. Their decision to not revisit that same domain at a later time may be part of a strategy for scaling their infrastructure to the large number of targets appearing in their CT logs.

Figure 5 shows the distribution of the number of requests sent by each visitor for all three kinds of subdomains we register. We find that almost 80% of all visitors to `Impersonating` domains send "single-shot" requests, or a single request without returning, while that same behavior is only observed from approximately 30% of visitors to `Baseline` and `Sensitive` domains. Overall, we discover that the vast majority of visitors send fewer than 10 requests to our honeypots, with the exception of a small number of visitors that send thousands of requests. For these visitors, we find that they typically send the same number of requests to each of our newly registered domains, indicating the use of a pre-curated scan-list – such as one associated with a scanning tool.

This observed behavior demonstrates the differences between bots targeting domains within these two groups, with bots operated by anti-phishing organizations monitoring for newly-created phishing websites making a small number of requests shortly after certificate creation in order to detect malicious content. Meanwhile, bots targeting domains that indicate potentially vulnerable systems initiate more requests for a much longer period of time, probing for vulnerabilities.

Bot Distributions

Figure 6 shows the top countries from which CT bots originated from in our dataset. Currently, there is an order of magnitude more CT bots in the United States than there are in any other country. Interestingly, we find that bots that visited `Impersonating` domains are found in each of the top countries, while there exists a number of countries that do not host any bots that visited `Sensitive` or `Baseline` domains (such as Germany and the UK). While the difference in total unique IP addresses from each domain explains these gaps, it also shows the clustering of these bots in a fraction of all represented countries.

To further understand the origins of CT bots, we utilized the IpInfo [23] API to determine the Autonomous System (AS) type of each bot IP. We present our findings in Figure 7. We find that the proportion of each IP address type is consistent across all three domain categories. That is, the majority of CT bots originate from hosting providers. Interestingly, the second-largest group of CT bots originate from ISP address ranges. This is an unexpected result, and indicates that many CT bots are either running on infected home devices (i.e., IoT devices) or are using ISP-based proxy services to evade detection [62].

IP Blocklist Presence

To determine the reputation of CT bot IP addresses, we utilized a subset of the IP blocklists provided by Firehol [17] that correspond to bot behavior. In total, we searched for IP addresses on 43 blocklists, a full breakdown of which are listed in Table 6 of the Appendix. Of the 31,898 unique IP addresses that sent requests to CTPOT, 4.5% appeared on at least one IP blocklist during our data collection period. This corresponds to 5.1% `Impersonating`, 6.6% `Sensitive`, and 7.7% `Baseline` IPs.

Figure 8 shows the overlap of CT bot IP addresses that appear on at least one IP blocklist, based on the domain types they sent requests to. Compared to Figure 3, we find a much larger overlap among maliciously labeled IP addresses. This

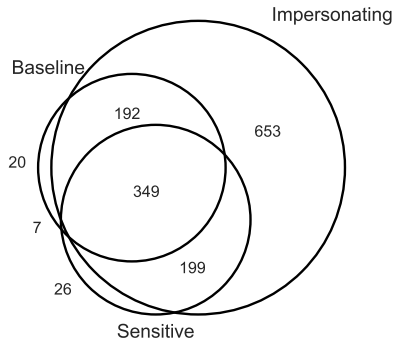

Figure 8: *Distribution of malicious bot IP addresses in our dataset by domain-type targeted. We consider an IP address to be malicious if it appears on at least one IP blocklist.*

implies that attackers utilizing CT for target discovery cast a wide net by scanning domains of various kinds, rather than targeting a particular category of domain. However, similarly to the overall distribution of CT bot IP addresses, those that only visited `Impersonating` domains are the single-largest group of IP addresses in our dataset.

4.2 CT Bot Behavior

As we have shown, there exists distinct populations of CT bots that interact with only a specific subset of domains. The filtering of domain content prior to interaction implies that each group has explicit goals in their use of CT. We now use the varying behaviors of bots in these groups to determine their intentions, including examining the extent to which attackers utilize CT to discover new targets and where they lie within these overall bot populations.

4.2.1 Request Targets

As the only information advertised about our measurement nodes is each domain through CT, studying requested file paths is a useful method to determine the intentions of a particular visitor. Table 2 shows the top five paths requested from each of the domain groups by the percentage of unique IP addresses (within each group) that requested that path at least once. Unsurprisingly, over 96% of IPs in each domain group request the website's root path, with this request occurring as a visitor's first query 96.6% of the time.

Beyond the domain root, we observed 24.1% of unique IP addresses queried for at least one additional file. The most common of these is for the file *favicon.ico*. This request is common if the client is using a real web browser, as this file is automatically requested by major web browsers in order to render a tab bar image [69]. Using these requests, we can begin to infer the number of CT bots that utilize a real browser to perform network requests. Doing so, we find only 11.9% of CT bots targeting `Impersonating` domains requested the site's favicon file. This is less than half as frequent as the bots targeting `Sensitive` and `Baseline` domains. We note however, that it is trivial for bots to initiate a request for *favicon.ico* in order to appear as though they are utilizing a real web browser. Therefore, we treat requests for this file as an upper bound on real browsers in our dataset, and rely on further analysis later in this section to determine the extent in which bots misidentify themselves.

Table 2: *Top paths requested by visitors to each domain type, by percentage of visitors that requested each path.*

Impersonating		Sensitive		Baseline	
Path	**Ratio**	**Path**	**Ratio**	**Path**	**Ratio**
/	97.27%	/	96.86%	/	97.79%
favicon.ico	11.96%	favicon.ico	25.08%	favicon.ico	28.55%
robots.txt	1.75%	robots.txt	3.60%	robots.txt	4.92%
my-account/	1.01%	login.php	2.91%	login.php	0.84%
US/US_aaiphh	1.00%	wp-login.php	0.85%	wp-login.php	0.69%
walmart-mo/	0.97%	.git/HEAD	0.33%	.git/HEAD	0.47%

Along with the *favicon.ico* file, all three domain groups receive requests for the file *robots.txt*. This file is commonly used by websites to control the requests made by bots. It contains access rules specified by the website owner to be followed by all web bots visiting the site, including which paths are permitted to be accessed, and which are not–including revoking all access to bots [22]. Ideally, this file is supposed to be read and followed by all web bots, however this is not always the case [72]. Over the course of our data collection period, we find that only a small fraction of bots send requests for the *robots.txt* file. Furthermore, fewer visitors request this file in the `Impersonating` and `Sensitive` domain groups than the `Baseline` group, with visitors in the `Impersonating` group requesting it the least often.

Past the top three most common, paths in each group begin to diverge. Here, we see requests towards `Impersonating` domains focused on common website endpoints such as *my-account/*. Meanwhile, requests towards the remaining two groups are towards endpoints such as *login.php* and *.git/HEAD*, which are commonplace in vulnerability scans [58].

POST Requests

In addition to the paths requested by each CT bot, we can determine malicious intent by observing POST requests to the domains we advertise. We assert that sending unsolicited POST requests to newly-created web endpoints is an overtly malicious action, regardless of the path to which the request was destined. Upon searching our dataset, we find that only 4,555 of all requests towards our measurement nodes were POST requests. This corresponds to 2,818 `Impersonating`, 901 `Sensitive`, and 836 `Baseline` requests. While this is a small fraction of the overall number of requests directed towards our measurement nodes, we reason that this could be a result of our endpoints responding to all requests with the same simple HTML message. As this content does not indicate the presence of a form to submit data to, bots simply choose to move on, instead of blindly sending POST requests.

Table 3: *Types of HTTP User-Agents encountered by measurement nodes during data collection period as a fraction of unique IP addresses in each category.*

User-Agent Type	Impersonating	Sensitive	Baseline
Browser	84.71%	78.38%	76.11%
Academic/Industry	5.64%	13.44%	15.10%
Library	3.90%	1.79%	4.31%
Scanning Tool	3.01%	3.50%	3.22%
Other	2.73%	2.88%	1.23%
Payload	0.01%	0.01%	0.03%

Table 4: *Types of TLS stacks encountered by measurement nodes during data collection period as a fraction of unique IP addresses in each category.*

Fingerprint Type	Impersonating	Sensitive	Baseline
Library	40.94%	17.58%	12.07%
Academic/Industry	20.37%	30.63%	33.45%
Unknown	14.46%	20.46%	25.70%
Scanning Tool	12.52%	28.22%	26.59%
Browser	11.59%	2.94%	1.98%
Empty	0.12%	0.17%	0.21%

4.3 CT Bot Self-Identification

The HTTP User-Agent is the web's primary method for self-identification of clients. It is standard practice for web bots to use this string to identify themselves to each web server, along with information about their purpose. In total, we recorded 1,746 unique User-Agents from CT bots during our data collection period. To better understand the behaviors of CT bots, we manually inspected each User-Agent we received and produced a label to each depicting the type of client it implies.

Table 3 shows the types of User-Agents our measurement nodes encountered from bots requesting subdomains from each of the three groups. We observed User-Agents from the following general groups: web browsers (e.g., Google Chrome and Firefox), network request libraries of programming languages (e.g., Python Requests library, and Go-http-client), academic or industry scanning tools (e.g., Googlebot and LinkedInBot), web scraping tools (e.g., wget and curl), malicious payloads (e.g., Log4j and code injection), and other strings we could not classify into one of the defined groups such as custom strings and messages.

We find that a majority of bots present a User-Agent of a browser. Furthermore, bots that interact with `Impersonating` domains are more likely to use the User-Agent of a browser than bots that interact with `Sensitive` and `Baseline` domains. We hypothesize that this behavior is because of the entities commonly operating the websites in each domain category. As malicious actors are more likely to create websites with impersonating domain names, anti-phishing organizations do not disclose their identities in order to prevent cloaking by attackers [59, 73]. Meanwhile, `Sensitive` and `Baseline` domains are likely operated by legitimate, benign entities who are unlikely to cloak any content on their sites, leading to approximately 10% more Academic/Industry User-Agents presented to those domains.

TLS Fingerprinting

Based on the HTTP User-Agents recorded by our measurement nodes, the majority of CT bots claim to use real browsers to perform network requests. However, it is known that web bots are likely to spoof their User-Agents to prevent detection and subsequent blocking by the websites they visit. To investigate the rate in which CT bots spoof their User-Agents, we utilize the JA3 TLS fingerprints we recorded from each bot that sent HTTPS requests to our measurement nodes, as described in Section 3.

In total, we recorded 113 unique JA3 TLS fingerprints from the clients interacting with our measurement nodes. Of these clients, 19.5% of all unique IP addresses presented more User-Agents than JA3 TLS fingerprints, with that corresponding to 19.4% of `Impersonating`, 12.3% of `Sensitive`, and 11.7% of `Baseline` visitors. This implies these bots changed their User-Agents between requests, while the underlying client remained the same. To ensure that IP-address churn was not the cause of this discrepancy, we calculated the length of each request session from bots towards the domains we leaked on CT, finding that 92% of all sessions last less than 10 minutes. This is far shorter than the average time IP address leases are renewed by major ISPs [63].

To identify the underlying clients making these requests, we query for all fingerprints in the *ja3er.com* [24] fingerprint database which maps JA3 hash values to observed HTTP User-Agents with those fingerprints. For each fingerprint we manually create a label based on the User-Agents returned from the database. However, for many JA3 fingerprint values, this database contains multiple HTTP User-Agents. In these cases, we produce a label based on the "lowest-level" User-Agent available. That is, if a particular JA3 fingerprint maps to the User-Agents of a browser *and* a network request library, we label that fingerprint as belonging to a network request library. We reason that a bot utilizing a network request library (e.g., Python Requests library) would benefit from reporting as a web browser as part of its detection-evasion strategy. However, a bot utilizing a web browser to send requests is unlikely to identify as a network-request library as there is nothing to be gained by this misidentification.

Using this methodology, we labeled each JA3 TLS fingerprint we received and present our results in Table 4. We observe a vastly different distribution of clients compared to what was reported by each CT bot. As we expect, the percentage of bots that utilize network request libraries to send requests is much greater in reality. Moreover, the number of bots that used a real web browser to send requests is only 2–11% based on their TLS fingerprint, compared to the 75–85% based on the reported User-Agents. This also shows how bots targeting `Impersonating` domains appear to take

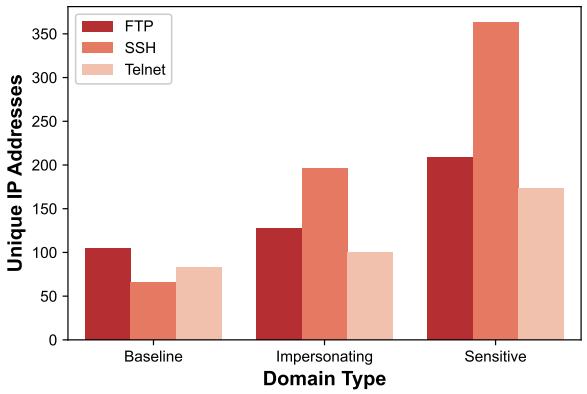

Figure 9: *Number of unique IP addresses that interacted with a network service honeypot in addition to sending a web request to one of the three domain groups.*

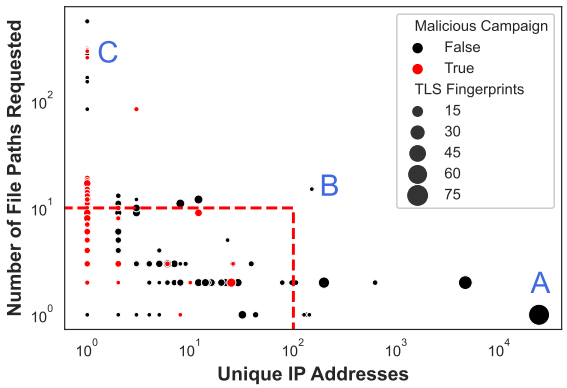

Figure 10: *Unique IP addresses encountered sharing the same subset of paths requested. Each marker represents one unique set of paths, with the size of each marker representing the number of TLS fingerprints encountered from the IP addresses which requested those paths. Red markers indicate that the cluster contains malicious requests. The bounding box with a corner at* $(x, y) = (10^2, 10^1)$ *contains 68.4% of all malicious bot campaigns in our dataset.*

more action to prevent fingerprinting than bots targeting `Sensitive` and `Baseline` domains, as approximately 11% more of these bots use a real browser.

Comparing the TLS fingerprints of each bot with the file paths requested in Table 2, we see that requests for *favicon.ico* are in line with the percentage of bots with a web browser TLS fingerprint. However, this is not the case for bots that requested `Sensitive` and `Baseline` domains, where over 25% of the bots that queried for *favicon.ico* did not have a TLS fingerprint that matched a known web browser. This indicates that bots in these categories could be using scanning tools or scanning lists that make requests for this file as a method to bypass bot detection.

These findings demonstrate that CT bots are likely to spoof their User-Agents to prevent detection. TLS fingerprinting remains a powerful tool for uncovering the true identities of these bots. However, we find that the *ja3er.com* fingerprint database is incomplete as 15–25% of our collected fingerprints were unknown. Increased usage of TLS fingerprinting and contributions to databases such as this will help improve the overall visibility into the clients utilized by bots. To assist in this, we submitted all User-Agents we received from the bots in our dataset to this database.

4.4 Network Service Interactions

As mentioned in Section 3, in addition to hosting a reverse proxy web server, each measurement node also hosted three low-interaction honeypot processes: SSH, FTP, and Telnet. These three services log all network interactions with visitors, but do not permit any authentication. This allows us to infer malicious intent among CT bots that not only request web content after observing a domain on a CT log, but also try to probe additional ports on the measurement nodes and even attempt to gain access to these additional services. For the purposes of this work, we consider any kind of interaction with a honeypot as malicious.

We correlate web request logs with honeypot logs using the client IP address that sent requests to each. Specifically, we associate a honeypot interaction with the closest web request temporally (i.e., an SSH interaction is associated to an `Impersonating` web request if that web request occurred the closest to that interaction compared to all other web requests from that host). In total, our measurement nodes received 22,839 network requests directed towards the ports of the three honeypot processes from 746 unique IP addresses. Of these, 675 (90.5%) went beyond simple network probes and attempted to authenticate with the particular service. Figure 9 shows the number of honeypot interactions that occurred from CT bots that sent web requests to domains of each category. We observe that CT bots that target `Sensitive` domains are more likely to attempt to interact with additional services on the host, compared to CT bots targeting domains of the other two groups. This shows that CT bots targeting such domains have malicious intentions in doing so, as these domains indicate the creation of a potentially vulnerable online service. Attempting to authenticate to these servers via Telnet, SSH, and FTP indicates the desire to achieve remote-command execution capabilities and acquire sensitive information.

4.5 CT Bot Campaign Analysis

It is common practice for bots to be used in scanning "campaigns", or scanning sessions where a pre-determined set of probes are sent to each encountered web host. Therefore, if the same bot software is deployed on many different machines, the file paths requested by each unique IP address would be the same. We use this expected behavior to identify connections between bots in our dataset. Specifically, we analyze the intersection of the total file paths requested by each unique IP address with all other IP addresses in our dataset. Additionally,

to further attribute the requests of multiple IP addresses to the same bot software, we analyze the JA3 TLS fingerprints of IP addresses that requested each set of file paths.

In total, bots in our dataset cluster into 539 unique sets of requested file paths. For the purposes of this analysis, we call each of these sets a distinct "campaign". Figure 10 shows the distribution of these campaigns, with the size of each marker indicating the number of TLS fingerprints encountered by bots that requested those specific paths. By analyzing the placements and sizes of each marker in this figure, we can create connections between seemingly isolated IP addresses, and identify the most active clusters within the CT bot population.

We observe three general trends in the characteristics of CT bot campaigns. The most populous cluster in our dataset (A) is comprised of bots that only requested the root file path. This cluster contains over 24K IP addresses, with 88 fingerprints shared between them. As the number of TLS fingerprints is large, with a small file path set, little can be deduced about the connections between bots in clusters such as this.

We can be more confident in the association between distinct IP addresses if either the number of paths requested is large enough, or if the number of IP addresses in that cluster is large with only one or a small number of TLS fingerprints shared among them. For instance, we observe one campaign (B) where 151 unique IP addresses requested the same 15 file paths, all while possessing the same TLS fingerprint. Similarly, we see a group of campaigns (C) that have 100–200 distinct file paths requested, with significant overlap in the specific paths requested between them. In cases such as these, we can deduce connections between distinct IP addresses that appear to be utilizing the same or similar bot software and target request lists.

Malicious Request Fingerprinting

We have discovered that bots in our dataset originating from unique IP addresses can be associated to the same campaign because of the requests they make. These campaigns vary in the number of files requested, and the number of bots that participate. We now seek to determine the effect of domain content on the volume of malicious activity directed towards hosts online, and the overall maliciousness of the campaigns identified in our dataset. To do this, we cross-reference all requested file paths in our dataset with the malicious bot traffic dataset curated by Li et al. [58]. This dataset contains labels for over 14K bot web requests performing common malicious actions (e.g., exploit strings and data exfiltration).

Figure 11 shows the malicious requests directed towards each of our three domain categories, by the number of unique IP addresses that executed them. During our data collection period, we encountered malicious requests in the form of data exfiltration, fingerprinting, and vulnerability exploitation, originating from 272 unique IP addresses. We note that the bot traffic dataset from Li et al. contains labels for other kinds of malicious requests (i.e., backdoor creation), but the malicious activity observed from CT bots was limited to these three categories. Of the three domain categories measured,

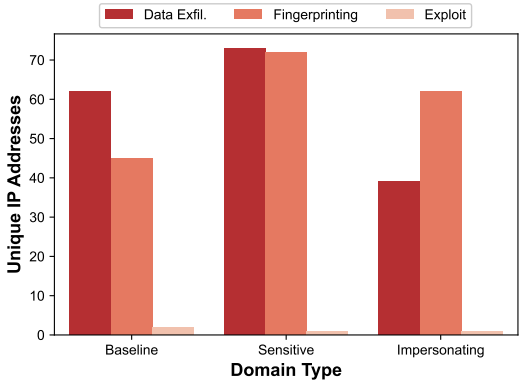

Figure 11: *Unique IP addresses in each domain group that requested a file path indicating one of the recognized malicious actions.*

`Sensitive` domains encountered the greatest number of bots that executed at least one malicious request, with data exfiltration being the most popular malicious action performed. As these domains indicate the presence of a potentially vulnerable web application, malicious bots may assume there could be sensitive information that is publicly available.

In addition to analyzing malicious activity on an IP-level, we also use this dataset to determine the overall maliciousness of the CT bot campaigns we identified. Of the 539 unique sets of requested file paths, we find that 105 contain at least one malicious request, depicted as the red markers in Figure 10. The majority of malicious campaigns have fewer than 10 requests and are shared between fewer than 100 IP addresses, with 68.4% falling within the bounding box with a corner at $(x,y) = (10^2, 10^1)$. This finding demonstrates the stark contrast in the malicious and benign uses of CT, where benign bots participate in large-scale scanning campaigns across many unique IP addresses, and malicious bots attempt to decrease their footprint by limiting the scope of their campaigns.

In conjunction with our findings in Section 4.1, the clear presence of malicious actors in the CT bot ecosystem emphasizes the necessity of website administrators to ensure the process of setting up a new web service concludes with certificate creation, after all security mechanisms are in place and tested. Due to CT logs, the seemingly innocuous action of creating a TLS certificate now publicly announces the existence of a new website, leading to the immediate attention of web bots, some with malicious intentions.

5 Case Studies

Most Active Bot

Of the 31,898 total unique IP addresses that visited our measurement nodes throughout our data collection period, only three exceeded 100K total requests. The most active of these (IP address: 3.85.226.XXX) sent over 327K requests, all targeting `Impersonating` domains that contained the strings "Coinbase" or "Alipay". Note that there has been an increasing number of phishing attacks against online cryptocurrency

exchanges [42, 55, 64] which could motivate new generations of CT bots searching for impersonating websites containing these popular trademarks. These requests only occurred between November 3rd, 2021 and November 22nd, 2021. On average, this bot sent approximately 30K requests per day, and over 36K requests per domain.

We observe two large spikes in incoming traffic to our measurement nodes. The first of these occurred on November 4th and 5th, and the second from November 8th through 10th. These two spikes can be directly attributed to this one particular bot as 83.2% of all requests on these dates originated from this IP address.

Regardless of the number of requests sent to each domain, they were all identical: GET requests over HTTPS to the root path of the domain. Moreover, requests to each domain occurred shortly after the certificate was registered and lasted for a short period of time, ranging from a few hours to days. Since this client focused on impersonating domains and did not appear on any blocklist during our data collection period, we hypothesize that this behavior is due to a poorly configured anti-phishing bot that was taken down upon realization of the request volume produced.

Log4J Exploitation Attempts

In December 2021, security researchers from Alibaba discovered a vulnerability in the Apache Log4J logging library. This vulnerability allowed attackers to remotely execute arbitrary code on the victim's machine using nothing more than loggable web events [28]. Attackers simply need to include JNDI or LDAP command strings in common web request fields such as requested paths and HTTP headers. If these fields are logged with an un-patched version of Apache Log4j, the included payloads will be executed on the victim's machine. Upon discovery of this vulnerability, websites across the Internet reported receiving thousands of exploitation attempts, regardless of the site's actual usage of Apache Log4j [27]. This implies early exploitation attempts of this novel vulnerability were conducted with Internet-wide scanning, agnostic to each site's content.

Since this vulnerability was released and mass-weaponized during our data-collection period, we received exploitation attempts on our measurement nodes. We analyzed our collected data to understand the use of CT to gather targets for this attack. In total, we received 2,188 total requests containing a Log4J payload in the HTTP headers and queried filepaths, originating from 183 unique IP addresses. These requests were directed towards both the specific domains we advertised on CT as well as the IP addresses of our measurement nodes directly. Our measurement nodes received 686 HTTP requests containing Log4J payloads towards the domains we advertised on CT, from 3 unique IP addresses. Conversely, our measurement nodes received an additional 1,502 Log4J requests towards their IP addresses (68.6% of all Log4J requests), from 181 unique IP addresses. This includes one IP address that sent Log4J payloads to the domains we advertised as well as the IP addresses of our measurement nodes.

Immediately, we observe that targeting specific domains advertised on CT is a much less popular strategy among attackers compared to Internet-wide scanning. However, upon analyzing the timestamps of each request, we see that requests towards the IP addresses of our measurement nodes started to occur on December 10, the date in which details regarding the Log4j vulnerability were made public. Meanwhile, such requests towards advertised domains did not begin until December 12, indicating a shift in strategy after initial Internet-wide scans ran their course.

Looking closer at the requests sent to our domains advertised on CT, we see that domains of the three categories received roughly the same number of requests, with `Impersonating`, `Sensitive`, and `Baseline` domains receiving 237, 221, and 228 requests, respectively. Of the three unique IP addresses that sent these requests, we found that two of the three belong to anti-abuse entities, with the remaining IP belonging to a machine in Microsoft's AS. Exploit payloads in these requests reside in 28 different HTTP headers along with the requested path. Additionally, all payloads appear to simply reach out to a unique subdomain of the primary domain *log4jdns.x00.it* over DNS, suggesting that these are simply probes to determine if a site is vulnerable. On the other hand, analysis of payloads sent to the IP address of each measurement node reveals that the majority of requests attempt to download code from an unknown server, with 20.9% of such requests being base-64 encoded shell scripts.

Our findings demonstrate that weaponization of Log4j exploits is currently limited to Internet-wide scans, rather than targeted attacks on domains publicized on CT. Together with our earlier findings regarding the small overlap between CT bots and Internet-scanning bots (Section 4.1), this suggests that the parties operating CT bots are different from those behind Internet-wide scanners with the former acting more benign than the latter.

6 Discussion and Future Work

6.1 Key Takeaways

- **Malicious CT Bots:** Our results show that when one creates a website, they must ensure that all security best practices are applied prior to creating TLS certificates. Once a domain appears on CT logs, admins should expect to receive numerous requests to their sites within minutes of certificate creation, from potentially malicious web bots. This is especially true for `Sensitive` domains that indicate the site could be a vulnerable web application, which are likely to receive tens of probes ranging from fingerprinting attempts to unsolicited POST requests. In total, we observe 105 malicious web-request campaigns targeting our measurement nodes. Furthermore, we find hundreds of unique IP addresses that extend their probes beyond web servers, attempting to authenticate with exposed network services such as SSH.

- **Bot Ecosystem Diversity:** Because of the rapid growth of the web, the population of bots interacting with websites is only becoming larger and more diverse. In this paper, we studied a previously unexplored subset of this ecosystem, demonstrating that CT bots are indeed distinct in source and behavior from traditional IP-based bots. Moreover, we showed that this population of CT bots can be further sub-divided into groups with distinct behaviors.
- **CT Bot Fingerprinting:** CTPOT is a highly versatile system, allowing defenders to find both the types of targets that bots are interested in (so they can be better defended), as well as malicious bots themselves. We have shown that using bot fingerprinting techniques, we can make connections between otherwise isolated bot requests. In total, we discovered 539 campaigns in our dataset, with many spanning across multiple unique IP addresses. By collecting IP addresses and fingerprints of malicious bots attracted to the CTPOT infrastructure, defenders can identify pools of IP addresses utilized by attackers and curate extensive blocklists to better protect newly-created websites from malicious requests. Continual identification of malicious bots effectively "drains" the IP address pools utilized by attackers, reducing the effectiveness of rapid IP address changes.
- **Malicious Use of CTPOT:** Malicious actors seeking to prevent detection and subsequent removal of their content could use a system such as CTPOT, along with similar fingerprinting techniques, to create real-time blocklists of bots operated by anti-abuse organizations. These lists could be fed directly into attacker infrastructures to ensure only real victims observe malicious content, such as phishing webpages and malware downloads. Operators of anti-abuse bots should be aware of this possibility and design probing strategies that reduce the chance of trivial evasion by attackers. For instance, utilizing real browsers to initiate requests and reducing the number of hosts used to conduct scans on a particular website would raise the bar for attacker evasion.

6.2 Limitations

Our analysis should be considered alongside certain limitations. As CTPOT was designed to study a large and diverse population of bots, the content we chose to serve to visitors is generic and simple, not conforming to any particular content-group. We therefore expect that some bots who initiated requests towards the domains we advertised on CT chose not to perform normal probing scans because of the content received from the web server. Rather, if we for instance hosted a full WordPress site on all `Sensitive` domains, we would likely receive more malicious requests such as fingerprinting or credential brute-forcing attempts.

Additionally, as this is a prototype study to introduce the community to this previously unexplored subset of web bots, our domain-generation strategy revolved around a limited set of categories and related strings. As mentioned in Section 3.1.1, we chose strings related to each of the three studied categories based on their popularity. However, we found that this can sometimes not be sufficient because of the rapid evolution of web. For instance, roughly halfway through our data collection period, the Log4J vulnerability targeting software written in Java was discovered. However, we did not have any subdomain strings containing either Java or Log4J in our `Sensitive` category. To ensure the integrity of our curated dataset, we did not modify the parameters of our setup during the entirety of our data collection period. However, future uses of CTPOT would be best-served to maintain a dynamic list of subdomain strings to remain on top of evolving trends.

We also note that our choice of TLD could bias our findings to bots that are only interested these low-cost domains. Our pilot study (Section 3.2) demonstrated that, in addition to an overall greater amount of traffic generated towards our measurement node with a low-cost TLD, there is a large overlap of interest in low and high cost TLDs. By limiting our study to a single TLD, we were able to narrow its scope, providing a more focused analysis on the effect of subdomain content on web bot behavior. We encourage future work to expand upon our pilot study, to fully understand the effects of TLD choice on CT bot behavior.

7 Related Work

Web Bot Detection & Measurement

The pervasiveness of bots in network communications has led to considerable work in developing strategies to detect and measure them. Prior studies have utilized supervised machine learning models to differentiate real users from web bots using features derived from their browsing patterns. This includes timing analysis of sessions (e.g. total visit time, time between requests, etc.), distribution of request types (GET, POST, HEAD, etc.), and the types of files requested [41, 61, 71]. Jacob et al. took this a step further by creating a system to not only detect and augment responses for particular bots, but also correlate requests from distinct IP addresses into coordinated campaigns [51]. Jan et al. addressed a common limitation of bot detection work, the lack of groundtruth datasets, by using data augmentation techniques to create artificial samples from real-world datasets [52].

Prior work has also explored web bots at scale to understand their general behaviors and strategies, including analyzing the web server logs of various academic websites to determine the common behaviors of search engine bots and general web bots [43, 44]. Xie et al. designed a bot detection system and used it to discover and analyze bots targeting a university campus network over the course of six months [74]. Li et al. created a honeysite system used to deploy various popular web application software to study the behaviors of bots targeting these particular applications. By ensuring the domains corresponding to these sites were never registered prior their

experiments, they were able to assume all visits to their honeysites were from web bots, rather than real users. Using this system, the authors were able to uncover malicious behaviors of bots targeting each web application, including credential brute forcing and exploitation attempts of known CVEs [58].

Existing work in the area of bot detection and characterization is focused primarily on Internet-wide scanning bots, or bots that feed off of pre-curated lists of long-standing websites. In this paper, we study a previously unexplored subset of bots, those which feed of CT logs. These logs provide bots with increased information about all sites, enumerating even otherwise-hidden subdomains that need to create a certificate.

Certificate Transparency

The creation and subsequent adoption of CT by the web has fostered the growth of an ecosystem consisting of numerous entities. Prior work has studied this ecosystem to determine how it operates in practice, including the actions of particular log operators, monitors, and auditors [49, 57, 65]. Stark et al. studied the overall success of the deployment of CT, to determine if its enforcement by browsers breaks web functionality [70].

Work has also been done to understand the security and privacy impacts of CT. This includes the implications of CT on end-user privacy as well as the unintended leakage of information because of the logging of all TLS certificates [47, 53, 66]. Dowling et al. also explore the resilience of CT against attackers at various vantage points using cryptographic means [45].

In our work, we explore an unintended use of CT: as a log of all newly created web endpoints. We therefore take inspiration from the work of Scheitle et al. who study the information leakage caused by CT [68]. The authors use CT to advertise random domains and monitor the network traffic directed towards the authoritative name servers for those domains, as well as network scans directed towards the IP addresses behind those domains. They find that DNS queries for domains appearing on CT logs reach their nameservers 73 seconds after certificate creation, and occasional port scans on their honeypot machines. Our work builds upon their findings by adding different experimental conditions (via the advertising of `Impersonating`, `Sensitive`, and `Baseline` domains) to assess the level of targeting, as well as modern fingerprinting techniques that allow us to uncover the real software utilized by CT bots and cluster seemingly independent clients as part of campaigns.

8 Conclusion

In this paper we presented CTPOT, a system for deploying and managing honeypots at various network vantage points which are associated with pseudo-random domain names advertised on Certificate Transparency (CT) logs. Using CTPOT, we conducted a ten week study of CT bots, recording 1.5 million total web requests from 31,898 unique IP addresses, directed towards the domains we advertised. As each domain generated by CTPOT was previously unused and completely unguessable, our curated dataset consists *entirely* of bot traffic.

By analyzing this dataset, we were able to uncover that CT bots are a distinct, yet highly-active, subset of the overall web bot population. During our data-collection period, we observed over ten times more requests from CT bots than IP-based bots. Moreover, our domain-generation strategy allowed us to observe unique behavior from CT bots targeting only a subset of domains on CT with particular content of interest. For instance, we discovered that 63% of unique IP addresses only targeted domains that impersonate popular trademarks. We also observed that bots targeting domains associated with common web applications exhibit substantially more malicious behavior, including attempts to authenticate with network services such as SSH occurring over twice as often as bots targeting other domains.

Lastly, we make associations between seemingly isolated bots, and find 539 request campaigns sharing identical file paths, and TLS fingerprints. Our findings demonstrate how operators deploy their bots into large infrastructures across many unique IP addresses. Furthermore, we identify malicious activity within these campaigns, discovering 17.6% contain payloads relating to data exfiltration, fingerprinting, and vulnerability exploitation.

Availability

Using CTPOT we curated a dataset consisting entirely of requests originating from CT bots. To assist in the understanding and further exploration of CT bots by the research community, we make our dataset available to other researchers at https://uninvited-guests.github.io.

Acknowledgements

We thank our shepherd Ralph Holz and the anonymous reviewers for their helpful feedback. 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] Anssi incident. https://www.mozilla.org/en-US/security/advisories/mfsa2013-117/.

[2] Apache traffic server. https://trafficserver.apache.org.

[3] Apache trafficserver ja3 fingerprinting plugin. https://docs.trafficserver.apache.org/en/9.1.x/admin-guide/plugins/ja3_fingerprint.en.html.

[4] Apple's certificate transparency policy. https://support.apple.com/en-gb/HT205280.

[5] Apwg phishing activity trends report - q3 2021. https://docs.apwg.org/reports/apwg_trends_report_q3_2021.pdf.

[6] Bind 9 nameserver. https://www.isc.org/bind/.

[7] Calidog certstream. https://certstream.calidog.io.

[8] Certbot. https://certbot.eff.org.

[9] Certificate transparency. https://certificate.transparency.dev.

[10] Certificate transparency - acm queue. https://queue.acm.org/detail.cfm?id=2668154.

[11] Chrome certificate transparency policies. https://chromium.googlesource.com/chromium/src/+/refs/heads/main/net/docs/certificate-transparency.md#Chrome-Policies.

[12] Comodo incident. https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html.

[13] Cowrie. https://www.cowrie.org.

[14] Diginotar hack. https://slate.com/technology/2016/12/how-the-2011-hack-of-diginotar-changed-the-internets-infrastructure.html.

[15] Facebook certificate transparency api. https://developers.facebook.com/docs/certificate-transparency-api/.

[16] Firefox - certificate transparency. https://developer.mozilla.org/en-US/docs/Web/Security/Certificate_Transparency.

[17] Firehol ip blocklists. https://github.com/firehol/blocklist-ipsets.

[18] Google certificate transparency sdk. https://github.com/google/certificate-transparency.

[19] Googlebot. https://developers.google.com/search/docs/advanced/crawling/googlebot.

[20] honeypot-ftp. https://github.com/alexbredo/honeypot-ftp.

[21] Https encryption on the web. https://transparencyreport.google.com/https/overview?hl=en.

[22] Introduction to robots.txt. https://developers.google.com/search/docs/advanced/robots/intro.

[23] Ipinfo. https://ipinfo.io.

[24] ja3er.com tls fingerprint database. https://ja3er.com.

[25] Let's encrypt rate limits. https://letsencrypt.org/docs/rate-limits/.

[26] Let's encrypt usage stats. https://letsencrypt.org/stats/.

[27] Log4j mass scanning. https://unit42.paloaltonetworks.com/apache-log4j-vulnerability-cve-2021-44228/#mass-scanning.

[28] Log4j vulnerability. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-44228.

[29] Mozilla - certificate transparency. https://wiki.mozilla.org/PKI:CT.

[30] Nginx. https://www.nginx.org.

[31] nitko. https://github.com/Tib3rius/nitko.

[32] Open sourcing JA3. https://engineering.salesforce.com/open-sourcing-ja3-92c9e53c3c41.

[33] Openphish. https://openphish.com/index.html.

[34] Opera browser - misissued certificates. https://blogs.opera.com/security/2015/10/misissued-certificates/.

[35] Seleium. https://www.selenium.dev.

[36] Turktrust incident. https://blog.mozilla.org/security/2013/01/03/revoking-trust-in-two-turktrust-certficates/.

[37] wget. https://www.gnu.org/software/wget/.

[38] Yandexbot. https://yandex.com/support/webmaster/robot-workings/robot.html.

[39] Pieter Agten, Wouter Joosen, Frank Piessens, and Nick 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.

[40] Babak Amin Azad, Oleksii Starov, Pierre Laperdrix, and Nick Nikiforakis. Web runner 2049: Evaluating third-party anti-bot services. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 135–159. Springer, 2020.

[41] Andoena Balla, Athena Stassopoulou, and Marios D Dikaiakos. Real-time web crawler detection. In *2011 18th International Conference on Telecommunications*, pages 428–432. IEEE, 2011.

[42] Megan DeMatteo. Scammers Stole $14 Billion in Crypto in 2021. Here's How Investors Can Protect Their Coins. https://time.com/nextadvisor/investing/cryptocurrency/common-crypto-scams/, 2022.

[43] Marios D Dikaiakos, Athena Stassopoulou, and Loizos Papageorgiou. An investigation of web crawler behavior: characterization and metrics. *Computer Communications*, 28(8):880–897, 2005.

[44] Derek Doran, Kevin Morillo, and Swapna S Gokhale. A comparison of web robot and human requests. In *Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining*, pages 1374–1380, 2013.

[45] Benjamin Dowling, Felix Günther, Udyani Herath, and Douglas Stebila. Secure logging schemes and certificate transparency. In *European Symposium on Research in Computer Security*, pages 140–158. Springer, 2016.

[46] Arthur Drichel, Vincent Drury, Justus von Brandt, and Ulrike Meyer. Finding phish in a haystack: A pipeline for phishing classification on certificate transparency logs. In *The 16th International Conference on Availability, Reliability and Security*, pages 1–12, 2021.

[47] Saba Eskandarian, Eran Messeri, Joseph Bonneau, and Dan Boneh. Certificate transparency with privacy. *arXiv preprint arXiv:1703.02209*, 2017.

[48] Edona Fasllija, Hasan Ferit Enişer, and Bernd Prünster. Phish-hook: Detecting phishing certificates using certificate transparency logs. In *International Conference on Security and Privacy in Communication Systems*, pages 320–334. Springer, 2019.

[49] Josef Gustafsson, Gustaf Overier, Martin Arlitt, and Niklas Carlsson. A first look at the ct landscape: Certificate transparency logs in practice. In *International Conference on Passive and Active Network Measurement*, pages 87–99. Springer, 2017.

[50] Martin Husák, Milan Čermák, Tomáš Jirsík, and Pavel Čeleda. Https traffic analysis and client identification using passive ssl/tls fingerprinting. *EURASIP Journal on Information Security*, 2016(1):1–14, 2016.

[51] Gregoire Jacob, Engin Kirda, Christopher Kruegel, and Giovanni Vigna. {PUBCRAWL}: protecting users and businesses from crawlers. In *21st {USENIX} Security Symposium ({USENIX} Security 12)*, pages 507–522, 2012.

[52] Steve TK Jan, Qingying Hao, Tianrui Hu, Jiameng Pu, Sonal Oswal, Gang Wang, and Bimal Viswanath. Throwing darts in the dark? detecting bots with limited data using neural data augmentation. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1190–1206. IEEE, 2020.

[53] Daniel Kales, Olamide Omolola, and Sebastian Ramacher. Revisiting user privacy for certificate transparency. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 432–447. IEEE, 2019.

[54] Brian Kondracki, Babak Amin Azad, Oleksii Starov, and Nick Nikiforakis. Catching transparent phish: Analyzing and detecting mitm phishing toolkits. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 36–50, 2021.

[55] Brian Krebs. How Coinbase Phishers Steal One-Time Passwords . https://krebsonsecurity.com/2021/10/how-coinbase-phishers-steal-one-time-passwords/, 2021.

[56] Ben Laurie. Certificate transparency. *Communications of the ACM*, 57(10):40–46, 2014.

[57] Bingyu Li, Jingqiang Lin, Fengjun Li, Qiongxiao Wang, Qi Li, Jiwu Jing, and Congli Wang. Certificate transparency in the wild: Exploring the reliability of monitors. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2505–2520, 2019.

[58] Xigao Li, Babak Amin Azad, Amir Rahmati, and Nick Nikiforakis. Good bot, bad bot: Characterizing automated browsing activity. In *2021 IEEE symposium on security and privacy (sp)*, page 17, 2021.

[59] Zhou Li, Sumayah Alrwais, XiaoFeng Wang, and Eihal Alowaisheq. Hunting the red fox online: Understanding and detection of mass redirect-script injections. In *2014 IEEE Symposium on Security and Privacy*, pages 3–18. IEEE, 2014.

[60] He Liu, Kirill Levchenko, Márk Félegyházi, Christian Kreibich, Gregor Maier, Geoffrey M Voelker, and Stefan Savage. On the effects of registrar-level intervention. In *USENIX Workshop on Large-scale Exploits and Emergent Threats (LEET)*, 2011.

[61] Anália G Lourenço and Orlando O Belo. Catching web crawlers in the act. In *Proceedings of the 6th international Conference on Web Engineering*, pages 265–272, 2006.

[62] Xianghang Mi, Xuan Feng, Xiaojing Liao, Baojun Liu, XiaoFeng Wang, Feng Qian, Zhou Li, Sumayah Alrwais, Limin Sun, and Ying Liu. Resident evil: Understanding residential ip proxy as a dark service. In *2019 IEEE symposium on security and privacy (SP)*, pages 1185–1201. IEEE, 2019.

[63] Giovane CM Moura, Carlos Ganán, Qasim Lone, Payam Poursaied, Hadi Asghari, and Michel van Eeten. How dynamic is the isps address space? towards internet-wide

dhcp churn estimation. In *2015 IFIP Networking Conference (IFIP Networking)*, pages 1–9. IEEE, 2015.

[64] Emma Newbery. Thousands of Coinbase Users Hit by Phishing Attack – Here's How to Protect Yourself. https://www.fool.com/the-ascent/cryptocurrency/articles/thousands-of-coinbase-users-hit-by-phishing-attack-heres-how-to-protect-yourself/, 2021.

[65] Carl Nykvist, Linus Sjöström, Josef Gustafsson, and Niklas Carlsson. Server-side adoption of certificate transparency. In *International Conference on Passive and Active Network Measurement*, pages 186–199. Springer, 2018.

[66] Richard Roberts and Dave Levin. When certificate transparency is too transparent: Analyzing information leakage in https domain names. In *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society*, pages 87–92, 2019.

[67] Yuji Sakurai, Takuya Watanabe, Tetsuya Okuda, Mitsuaki Akiyama, and Tatsuya Mori. Discovering httpsified phishing websites using the tls certificates footprints. In *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 522–531. IEEE, 2020.

[68] Quirin Scheitle, Oliver Gasser, Theodor Nolte, Johanna Amann, Lexi Brent, Georg Carle, Ralph Holz, Thomas C Schmidt, and Matthias Wählisch. The rise of certificate transparency and its implications on the internet ecosystem. In *Proceedings of the Internet Measurement Conference 2018*, pages 343–349, 2018.

[69] Konstantinos Solomos, John Kristoff, Chris Kanich, and Jason Polakis. Tales of favicons and caches: Persistent tracking in modern browsers. In *Network and Distributed System Security Symposium*, 2021.

[70] Emily Stark, Ryan Sleevi, Rijad Muminovic, Devon O'Brien, Eran Messeri, Adrienne Porter Felt, Brendan McMillion, and Parisa Tabriz. Does certificate transparency break the web? measuring adoption and error rate. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 211–226. IEEE, 2019.

[71] Pang-Ning Tan and Vipin Kumar. Discovery of web robot sessions based on their navigational patterns. In *Intelligent Technologies for Information Analysis*, pages 193–222. Springer, 2004.

[72] Nikos Virvilis, Bart Vanautgaerden, and Oscar Serrano Serrano. Changing the game: The art of deceiving sophisticated attackers. In *2014 6th International Conference On Cyber Conflict (CyCon 2014)*, pages 87–97. IEEE, 2014.

[73] David Y Wang, Stefan Savage, and Geoffrey M Voelker. Cloak and dagger: dynamics of web search cloaking. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 477–490, 2011.

[74] Guowu Xie, Huy Hang, and Michalis Faloutsos. Scanner hunter: Understanding http scanning traffic. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pages 27–38, 2014.

[75] Penghui Zhang, Adam Oest, Haehyun Cho, Zhibo Sun, RC Johnson, Brad Wardman, Shaown Sarker, Alexandros Kapravelos, Tiffany Bao, Ruoyu Wang, et al. Crawlphish: Large-scale analysis of client-side cloaking techniques in phishing. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2021.

A Appendix

Table 5: *Strings used to construct domains for each of the three measurement categories.*

Impersonating	Sensitive	Baseline
apple	phpmyadmin	banana
paypal	wp-admin	pear
facebook	wordpress	peach
twitter	drupal	strawberry
instagram	admin	watermelon
linkedin	cpanel	melon
google	demo	cherry
youtube	testphp	grape
chase	administrator	grapefruit
venmo	mail	avocado
zoom	mailserver	blueberry
microsoft	mysql	coconut
dhl	postgres	clementine
bestbuy	db	fig
dropbox	sql	guava
youtube	ssh	honeydew
qq	webmin	kiwi
baidu	internal	lemon
sohu	private	mango
taobao	members	peach
yahoo	staging	artichoke
netflix	blog	asparagus
reddit	test	arugula
office	webmail	bean
alipay	wp	beet
coinbase	ws	broccoli
myshopify	example	cabbage
twitch	login	carrot
ebay	dev	cauliflower
adobe	demo	corn
aliexpress	ftp	cucumber
tiktok	gateway	ginger
alibaba	irc	kale
amazonaws	old	lettuce
spotify	smtp	mushroom
walmart	shop	pea
nih	files	pepper
americanexpress	app	potato
tmall	git	pumpkin
canva	hostmaster	red
bing	firewall	blue
tdameritrade	api	green
wellsfargo	store	orange
robinhood	printer	yellow
bankofamerica	users	purple
binance	proxy	violet
kraken	forum	indigo
gemini	vpn	brown
bittrex	m	black
indeed	ns	white
	Qualifiers	
reviews		
tutorials		
lessons		
assistance		
support		
help		
advice		
coaching		
education		
guidance		

Table 6: *List of Firehol blocklists used to determine maliciousness of CT bot IP addresses.*

Firehol Blocklist
Alienvault Reputation
Blocklist De Bots
Blocklist De Bruteforce
Blocklist De
Blocklist De SSH
Blocklist De Strongips
Blocklist Net Ua
Botscout 1d
Botscout 30d
Botscout 7d
Botscout
Botvrij Dst
Bruteforceblocker
Cleantalk 1d
Cleantalk 30d
Cleantalk 7d
Cleantalk New 1d
Cleantalk New 30d
Cleantalk New 7d
Cleantalk New
Cleantalk Updated 1d
Cleantalk Updated 30d
Cleantalk Updated 7d
Cleantalk Updated
Cruzit Web Attacks
Cta Cryptowall
Cybercrime
Dyndns Ponmocup
Et Compromised
Et Tor
Gpf Comics
Greensnow
Haley SSH
Malc0de
Myip
Php Dictionary 1d
Php Dictionary 30d
Php Dictionary 7d
Php Dictionary
Threatcrowd
Turris Greylist
Uscert Hidden Cobra
Vxvault