Ready or Not, Here I Come: Characterizing the Security of Prematurely-public Web Applications

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

Michael Ferdman
Stony Brook University
mferdman@cs.stonybrook.edu

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

Abstract—**Traditionally, the creation of a new web endpoint was seen as a private event, with its existence unknown to the outside world until deemed appropriate by the site owner. Indeed, the improbability of an attacker correctly predicting the exact address of a newly-created site allowed administrators sufficient time to configure their sites before users began to arrive. However, since the adoption of Certificate Transparency (CT), the act of obtaining a TLS certificate is announced to the public, where attackers can lie in wait for new targets to attack. This results in a new vulnerability period between the time that a site is issued a TLS certificate, and the time when administrators have finalized all security-related server configurations.**

In this paper, we present MAKO, a distributed web scanning system that determines the overall security posture of a host from a number of network vantage points. Using MAKO, we randomly sample 1% of all domains appearing on Certificate Transparency logs over 10 weeks, resulting in the auditing of 548,238 unique domains. By carefully and ethically analyzing the security posture of each host immediately upon discovery, as well as in the following hours to days, we are able to observe the change in their security posture over this time period and quantify the vulnerability window that attackers could exploit. Through this analysis, we discover 200,421 domains that increase their security posture in the time following their initial announcement on Certificate Transparency. Overall, our findings expose a downside of the Certificate Transparency system, where unknowing administrators prematurely announce the existence of their hosts before vital security measures are applied.

I. INTRODUCTION

The modern web heavily relies on the integrity and confidentiality guarantees of the HTTPS protocol. Today, adoption of HTTPS is higher than ever before, with over 79% of all websites utilizing it [1]. As a result, the creation and installation of TLS certificates has become a common, yet vital, step in the setup of a new website. Tools such as the Let's Encrypt Certbot [2] and Digicert's Certcentral [3] allow administrators to include certificate creation in their development workflows. Additionally, all-in-one web server software such as the Caddy Server [4] further simplify the process of deploying a web service by automatically generating and installing TLS certificates upon launch.

However, the seemingly benign action of creating a TLS certificate can have negative repercussions. Unlike other steps in the creation of an online service, such as host acquisition, the creation of a TLS certificate is a uniquely public action due to the Certificate Transparency (CT) system [5]. Any time a certificate is created or updated, an entry is appended to one or more public log files in nearly real-time. Due to the ubiquitous nature of the HTTPS protocol, these logs effectively serve as public announcements of all newly-created websites as they come online.

Prior work has characterized the volume and nature of traffic directed towards domains after the creation of TLS certificates [6]. The authors discovered that newly-created websites should expect requests from bots analyzing CT logs in as little as 12 seconds. Moreover, prior work found that a subset of these bots are malicious in nature, attempting actions such as data exfiltration, fingerprinting, and vulnerability exploitation. Overall, these findings highlight the need for website administrators to ensure that all security-related measures are in place *before* creating a TLS certificate. However, it is currently unclear what the security posture is of web hosts at the time their certificates are created.

While prior work has thoroughly explored overall trends in security posture among established websites [7]–[10], these rely on the assumption of a steady-state in the level of security of web hosts. However, it is likely that the resilience to attack of any particular online host will change in the hours or days after initial deployment. This includes server-side changes, such as, the configuration of network firewalls and content-based changes such as the enabling of access-control measures. Instances where administrators create TLS certificates prior to enabling these security measures provide attackers with a window of opportunity to exploit vulnerabilities on these sites *before* they become hardened to attack. Meanwhile, administrators may assume these vulnerable endpoints to be safe from attack due to a false sense of security derived from an endpoint's seemingly unguessable name or address. Thus, prompt scanning of domains appearing on CT logs can provide attackers with an additional pool of exploitable targets.

In this paper, we seek to fill the knowledge gap on this vulnerability window by quantifying the delta in the security posture of sites in the hours to days after TLS certificate creation. To do this, we develop MAKO, a web host security auditing system capable of quantifying the security posture of a networked host using a series of non-intrusive probes targeting various levels of the network stack. By conducting audit scans on domains sourced from CT logs, we can measure the security posture of sites on the Internet the moment they are publicly announced to the world. We then revisit these same sites periodically in the hours to days following their initial announcement, noting the delta in security posture over this interval.

Over the course of 10 weeks, we analyzed 548,238 unique domains, visiting each one four times over the week following its first appearance on a CT log, resulting in 2,347,840 total audit scans. By analyzing our curated dataset, we observed 200,421 hosts that improve their security posture in this time (i.e., their administrators were not done securing their hosts when their endpoints were announced on CT). For instance, we found 124,307 hosts that allowed direct access to sensitive network services such as SSH and database servers at first announcement, before closing these ports at a later time. Cases such as these demonstrate severe vulnerabilities that can arise from network administrators that create TLS certificates prematurely, without realizing the unintended side-effect of the CT system–the announcement of previously-hidden hosts and endpoints.

Our main contributions are as follows:

- We design and implement MAKO, a web host security auditing system that utilizes repeated scans with non-intrusive network probes to quantify the security-posture delta of sites over time.
- Using MAKO, we curate a dataset of web host security posture changes post TLS certificate creation. We draw attention to the new vulnerability window between the creation of a TLS certificate and the time it is hardened.
- Through a series of case studies, we demonstrate a dearth in knowledge amongst network administrators on the nature of the CT system, and the invisible side effects of obtaining TLS certificates.

II. MOTIVATION & BACKGROUND

Certificate Transparency

In the early 2010s, a series of Certificate Authority compromises led to the creation of TLS certificates for unauthorized entities [11]–[14]. These events resulted in the exploitation of many users through man-in-the-middle-attacks. In response to these events, the Certificate Transparency (CT) system was created to provide oversight into the actions of certificate authorities. This system is a series of public append-only logs of certificate creations, allowing the public to audit the actions of CAs, and for domain owners to monitor invalid certificate creations for their domains.

Most CT logs are managed by large organizations, although anyone can create a CT log and advertise its contents to the public. Similarly, submissions to CT logs can be made by anyone, but this process is typically done by CAs when creating a certificate for an entity. Each time a certificate is submitted to a CT log, the log provider returns a Signed Certificate Timestamp (SCT), which is then appended to the relevant certificate and used to validate the certificate's inclusion in a CT log. Participation in the CT system is enforced by web browsers, many of which require all certificates to possess an SCT, else the certificate will be treated as invalid [15]–[17]. This has resulted in over 90% of all CAs logging certificate creations to CT [18].

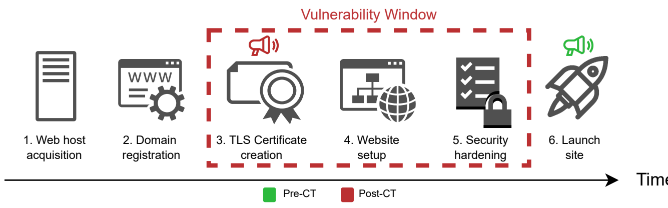

Fig. 1: The introduction of the Certificate Transparency (CT) system has introduced a vulnerability window for sites that prematurely create TLS certificates. Prior to CT, a new domain became public at the discretion of the site owner. This announcement now occurs the moment a certificate is created for a domain.

Auditing of CT logs can be easily accomplished through the use of readily-available APIs [19]–[21], allowing anyone to participate in the detection of invalid CA activity. However, the proliferation of the HTTPS protocol on the web [22], along with the almost total CA logging compliance, has resulted in these APIs providing a stream of almost all newly-created domains in real-time. This also includes Fully Qualified Domain Names (FQDNs) that would otherwise not have ever been discovered through traditional IP address crawling or domain guessing (e.g., 123abbf86.example.com), and top-level domain names (TLDs) from registries that do not share zone records with the public (e.g., country code TLDs).

CT has fundamentally changed the processes of creating a new website, illustrated in Figure 1. In the past, conventional wisdom suggested that a newly-created domain was unknown to the world, and thus, would not receive traffic until the site owner announces its existence in some way (e.g., advertising URLs). This allowed administrators the time to finalize the setup of web content as well as security hardening of the web server itself, before the new endpoint became known to the public. However, CT's logging of all TLS certificates, and the domains associated with those certificates, has shifted the *true* announcement of a new web endpoint to when its TLS certificate is generated. This has introduced a temporal gap between the perceived announcement time and the actual announcement time for a newly-created domain. Any vulnerability in the defenses of a web host that would be fixed prior to the perceived announcement time of the site can be exploited by attackers simply scanning CT logs for targets. The size of this vulnerability window, and the severity of vulnerabilities available to attackers however, is currently unknown.

Web Host Scanning

The ability to analyze web hosts and quantify their properties is invaluable to help understand the web's trends and weaknesses. Popular tools such as Nmap [23] and Zmap [24] allow users to identify the network configuration of a host and even the services executing on it. Furthermore, SDKs such as Python's Selenium [25] library allow users to create automated scanning scripts that utilize a real web browser to interact with web servers. Generally, automated browsing services such as these are referred to as "crawlers" or "bots".

Host scanning tools range in functionality, with the choice to use one over another being made based on the desired host information and constraints. The aforementioned Nmap is popular for its "quiet" operation, utilizing TCP probes to determine if a network port is publicly-accessible based on the response received. Meanwhile, comprehensive security tools, such as vulnerablilty scanners [26]–[28], provide detailed reports on the security posture of a website with a tradeoff of heavy request volume transmitted to the target host, potentially leading to detection and blocklisting.

Today, there are many services that provide data on Internet-wide trends through the use of host scanning. For instance, Censys [29] regularly conducts scans across the entire IPv4 space for the purpose of threat detection. Similarly, Shodan [30] provides a search engine of Internet-connected devices with data sourced from Internet crawlers.

It is also common for malicious activities to be performed using web bots, rather than manual human actions. One of the most important steps in an attack is target identification and reconnaissance. By leveraging automated host scanning, attackers can analyze large numbers of hosts and identify weaknesses in them. Many tools exist for this purpose, with one of the most popular being Metasploit [31]. These tools send a series of pre-crafted payloads that are used to identify and exploit known vulnerabilities.

III. MAKO: WEB HOST SECURITY AUDITING SYSTEM

To gauge the security posture of network hosts identified through the analysis of CT logs, we developed MAKO, a distributed web host security auditing system, illustrated in Figure 2. MAKO comprises two core components: a producer node that tails CT logs for candidate domains and worker nodes that probe each underlying web host. The distributed nature of MAKO emphasizes scalability as worker nodes can be freely added or removed to meet the current resource requirements of the overall system. Moreover, its stateless design allows for worker nodes to operate independently of each other, increasing MAKO's overall throughput.

A. Producer Nodes

Candidate domains are sourced by a producer node tailing CT logs with the Calidog Certstream API [21]. Due to the massive volume of certificates appearing on CT logs (10M+ per day), a variable sampling rate can be applied to limit the number of domains processed. For each certificate encountered, only fully-qualified domain names (FQDNs) are processed, with all wildcard domains discarded. Additionally, each domain (and corresponding IP address) is cross-referenced with a blocklist, allowing administrators the opportunity to opt-out of scanning. Each remaining domain is pushed onto a queue to be fetched by a worker node and analyzed.

The orchestration module is responsible for appending domains as they are encountered, and also scheduling re-crawls of all domains at set frequencies. As we seek to determine the vulnerability window of new sites as they appear on the CT logs, we re-crawl each domain at the frequencies of: one hour after first discovery, one day after first discovery, and one week after first discovery. This results in at least four data points collected for each discovered domain. Due to MAKO's stateless design, it is possible for a domain to be crawled more than four times if the site regenerates its certificate multiple times.

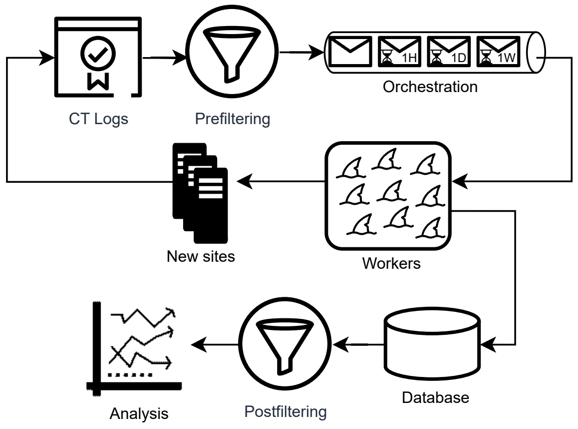

Fig. 2: *Architecture of* MAKO.

B. Worker Nodes

Analysis of each sampled domain is conducted by one of many worker nodes. Worker nodes are lightweight, with low resource requirements, allowing for a large number of workers to execute in tandem on a single machine. Moreover, worker nodes can be added or removed from the system in real-time to meet current demand.

1) Network Probes:

MAKO performs several types of network probes for all discovered sites. These probes seek to identify areas of weakened security posture, ranging from information disclosure to the use of outdated software vulnerable to public CVEs.

DNS Records

For each domain encountered, we send DNS queries for all associated `A`, `CAA`, and `TXT` records. Although we record all responses received from these requests, we designate the first IP address listed in the domain's `A` record as the primary IP address for that domain, and send all request packets to that IP for all subsequent probes. Note that, as the worker nodes are stateless, subsequent visits to the same domain may result in scans of different IP addresses, if the DNS records for the domain change between crawls.

Open Ports

To identify potentially sensitive services listening on a web host, we conduct a limited scan of 20 ports associated with popular network services (the list of ports and associated services is presented in Table II of the Appendix). For each port, we send a single TCP packet over a raw socket and record the resulting response payload or error code. For ethical reasons, and to increase the efficiency of MAKO, we do not attempt any further interaction with any web services encountered (such as authentication attempts). If a response is received, the socket is properly closed to free server resources.

Web Application Content

To determine the type of content served by a web server at the time of certificate creation, along with additional sources of information leakage, we collect the HTML code of the document root served by the web server, along with all returned HTTP headers.

Additionally, to discover hidden directories present on the web server, we send a series of HTTP GET requests for common directory names. We construct this list by first scanning the index HTML code, if available, and recording the top level directory name for all resources listed on the page. To supplement this, we queried the Github API [32] for the most common directory names in all repositories that contain HTML files. This list, along with additional directory names that are common to popular web applications, resulted in 103 directories to query in each scan (listed in Table IV of the Appendix).

For each request, we record the HTTP status code of the response. If a request is successful (i.e., 200 response code), we analyze the resulting HTML to determine if it is a directory index page produced by the web server. Such pages are commonly produced by default by web server software to make it easy for a user to browse the contents of the web server's file system. However this feature is recommended to be disabled by security experts, as it can reveal hidden directories containing sensitive information [33]. To determine if a response contains a directory indexing, we constructed a list of regular expressions that match the format of directory indexing pages for popular web server software. We show these regular expressions in Table III of the Appendix. If a regular expression matches at least one directory, a boolean flag is set to "TRUE", indicating the web server supports directory indexing at that time.

For all HTTP requests, we include a `User-Agent` header randomly-selected from a set of popular modern browsers. The random seed used to select this User-Agent is tied to the scanned domain, making all scans to the same domain use the same User-Agent header. Moreover, we include an `X-Experiment` header in all HTTP requests initiated by MAKO, which includes a link to a project website hosted by our institution. This website includes detailed information on the purpose of our study and contact information to direct opt-out requests.

C. Deployment and Data Collection

Using the described system, we deploy 8 data collection nodes located in our academic institution's datacenter, supporting a total of 400 MAKO worker processes. Due to the massive scale of certificate creations, we impose a 1% sampling rate on domains sourced from the CT logs. That is, for each certificate log entry encountered on CT, we parse out all FQDNs and sample from that list (i.e., we apply the sampling rate to the domains, not certificates).

We take great care in designing MAKO, and the experiments we perform with it, as to prevent any undue strain on the sites we analyze. We emphasize that MAKO only requests public information from each web host it encounters, and at no point do our probes attempt to exploit vulnerabilities or gain access to resources that are not available to any visitor to a site. Moreover, we limit the number of probes so that the number of requests MAKO sends is similar to the number of requests a single user produces when visiting a modern website [34].

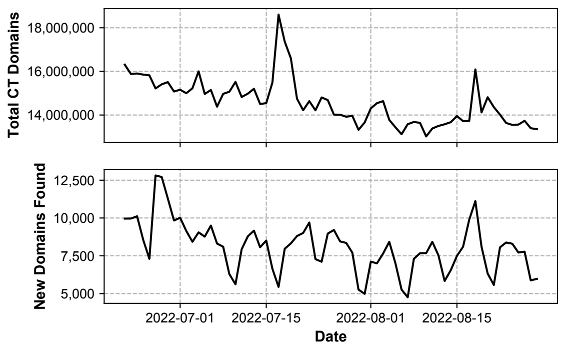

Fig. 3: *(Top) Total domains appearing on CT logs each day of our data collection period. (Bottom) Total new unique domains processed by* MAKO *post-filtering and sampling.*

IV. EXPERIMENTAL RESULTS

We report the findings of our deployment of MAKO, analyzing domains appearing on CT for 10 weeks, from June 22, 2022 to August 29, 2022. In total, we conducted 43,005,238 scans of 11,521,071 unique domains. However, CT logs contain entries for not only newly-created TLS certificates, but also for the renewals of existing TLS certificates. As we are only interested in the change in security posture of brand-new websites, we filter established sites from our dataset. To do this, we utilized the historic CT log database of crt.sh [35] to determine the earliest recorded TLS certificate for each domain in our dataset. Specifically, we recorded the timestamp on which each certificate was created and compared it with the timestamp of our earliest crawl of that site, discarding all domains in which a certificate was created at least one day before our first scan.

In addition to this filtering logic, we also discarded all domains in which MAKO did not complete the four planned auditing scans (original scan, followed by three recrawls described in Section III). These incomplete scans are the result of domains and network subnets being added to our blocklists before all visits could be completed. In our dataset, we found a small number of domains that were scanned by our crawlers more than four times. This is a result of domains renewing their TLS certificate multiple times during our data collection period.

We also remove all domains from our dataset that do not exhibit any web activity during our data collection period. Specifically, we first remove all domains in which no DNS `A` record is ever available. These domains generally result from large hosting providers that create temporary VMs for

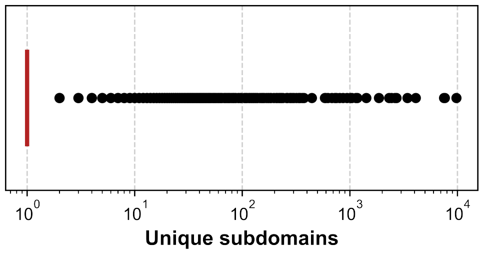

Fig. 4: *Distribution of the number of unique subdomains associated with each primary domain in our dataset.*

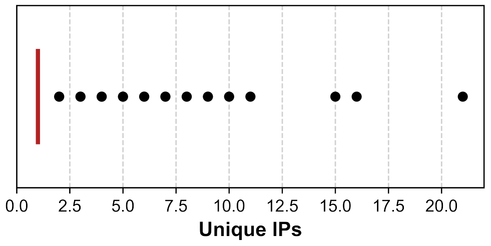

Fig. 6: *Distribution of total IP addresses encountered for each domain in our dataset.*

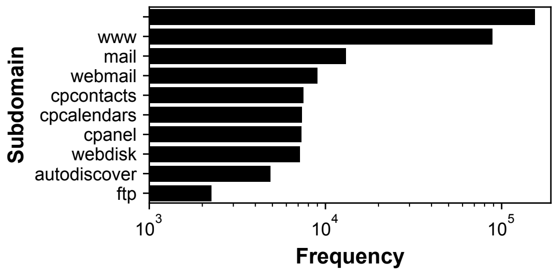

Fig. 5: *Most common subdomains associated with TLS certificates in our dataset.*

the purpose of certificate creation, before promptly tearing-down the VM. Similarly, we remove all domains in which each crawl shows ports 80 and 443 closed. While these sites may be hosting content on non-standard ports, we focus our study on hosts that eventually provide web-based content on the standard ports.

After applying the previously-described filtering steps, we are left with a dataset comprising 2,347,840 scans of 548,238 newly-created domains. Figure 3 (top) shows the total number of domains encountered on the CT logs each day (prior to filtering and sampling), with Figure 3 (bottom) showing the number of previously-unseen domains processed by MAKO on each day of our data collection period (after filtering and sampling). We observe a consistent number of new domains each day, with an exception of two spikes in activity at the beginning and end of our data collection period. Furthermore, we see an unusual increase in the total number of domains appearing on the CT logs on July 19, but due to our 1% sampling rate, this increase does not follow in our filtered dataset.

Figure 4 shows the distribution of the number of unique subdomains used in TLS certificates for each primary domain in our dataset. We find that 94% of all primary domains encountered created TLS certificates for only one subdomain. Due to our low sampling rate of the CT logs, we are unlikely to encounter additional certificate creations for a particular domain, unless that domain generates TLS certificates at an abnormally high rate. We see this in our dataset's outliers, where large entities, such as cloud and DNS providers, create TLS certificates for unique subdomains assigned to their clients.

Figure 5 shows the top 10 most popular subdomains in our dataset, by the number of unique primary domains in which the subdomain appears. We find that it is most common for a domain to not have a subdomain (e.g., `example.com`). Beyond this, we can see the large market share of cPanel in new website creations. The subdomains `webmail`, `cpcontacts`, `cpcalendars`, `cpanel`, `webdisk`, and `autodiscover` are all cPanel *Service Domains* [36]. These subdomains are all automatically created when a website is created using cPanel (the subdomain `autodiscover` is used by the Microsoft Outlook and Mozilla Thunderbird email clients to automatically discover and configure access to domain email servers [37]).

Additionally, we see potentially sensitive subdomains such as `mail` and `ftp` appear many times during our data collection period. Prior to the existence of CT, an attacker attempting to discover vulnerable endpoints would need to correctly predict the presence of these subdomains, generating noisy network traffic in the process. By simply ingesting CT logs, attackers can now observe certificates created for such endpoints and immediately direct probes towards them.

A. Web Host Security Posture Analysis

Using MAKO, we curated an extensive dataset consisting entirely of newly-created domains. As described in Section III, worker nodes probe each host from a number of network vantage points. Below, we share the results of each probe on the network stack, conducting differential analysis on the periodic scans of each domain over the hours to days after initial announcement on CT.

1) DNS Records: As described in Section III-B1, we consider the first IP address listed in each domain's `A` record to be the primary address for that domain, and send all subsequent probes to that address. Figure 6 shows the number of IP addresses encountered for each domain in our dataset. Overall, we find that 75.4% of all domains in our dataset have the same IP addresses throughout all crawls. Thus, an attacker that was able to gain access to the server (i.e., through exploiting a vulnerability or obtaining credentials) when observing a domain on the CT logs can "lie-in-wait" until the website is fully online before secretly exfiltrating sensitive information.

2) Port Scan: In support of web content, it is common for an online host to run other services, such as databases and mail servers, on auxiliary ports. However, as these services

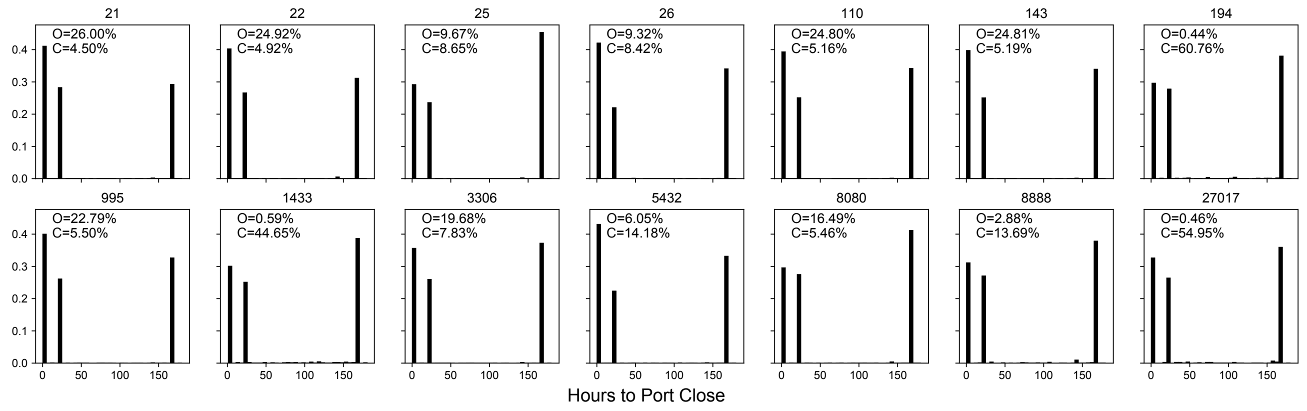

Fig. 7: *Distribution of the number of hours it took domains in our dataset to close specified ports. Percentages in top-left corner of each plot represent the fraction of hosts which initially had the specified port* ***O****pen when first encountered, and of those, what fraction* ***C****losed that port by the end of our data colelction period.*

can be highly sensitive, it is best practice to restrict access to them from the Internet using network firewalls. As described in Section III-B1, we conduct a TCP scan on 20 ports commonly used by sensitive network services. In total, we found 298,530 unique domains that had at least one sensitive service (excluding ports 80 and 443) publicly accessible on our first visit, with an average of 2.3 ports open. Furthermore, we observed 25,266 domains in which more than half of the scanned network services are publicly accessible on our first visit.

Figure 7 shows the percentage of all domains in our dataset that had one of the recorded sensitive ports publicly accessible during our first visit to it (O), and the percentage of those that closed that port at some point in the future (C). We noticed in our dataset a subset of hosts that appeared to close all scanned ports by the time of our last crawl. As we cannot be certain if this change occurred due to the hardening of site security, or if the host was completely decommissioned, we discard these hosts from our dataset for this analysis.

Each subplot shows the time distribution when these hosts closed the particular port. The majority of port closure events correspond to the three recrawl intervals MAKO schedules for each encountered domain, but a small number are in between, or after, these times, where a domain creates a new certificate in the hours or days after the original. For clarity, we restrict the axis of these plots to just over a week after first crawl, though a small fraction of outliers fall outside this range.

Ports for popular services such as FTP (21) and SSH (22) are the most commonly open among hosts in our dataset, at 26% and 24.9% of all domains having the port open at first crawl, respectively. However, as these services are utilized by administrators to remotely manage a website, less than 5% of these hosts close the two ports during the timeframe of our audit scans. Conversely, less popular, though highly-sensitive services such as MongoDB (27017) and Microsoft SQL Server (1433), are initially open by drastically fewer hosts, but hosts that run these services tend to quickly close these ports. Any period in which these services are open to the Internet presents the risk of them falling victim to the growing number of ransomware targeting NoSQL/SQL Server databases [38], [39].

We also observed varying behavior among hosts in the time period in which particular ports are closed. For instance, we found that port 194, commonly used to host IRC servers, is closed by 60.76% of hosts, but a majority of hosts that close this port do so more than a day after certificate creation. On the other hand, database software PostgreSQL (5432) is closed by the majority of hosts within the first hour after certificate creation. While this is significantly faster than other services, a quick attacker can still inflict substantial damage within the short timeframe.

Comparing these results to established hosts in our dataset, allows us to identify differences in behavior between newly-created websites and those that have been online for a longer period of time. These hosts, which we filtered from our dataset using the methodology described earlier, close their ports much less often during our data collection period than newly-created sites. For instance, ports 21 and 22 are closed on approximately 1% of established hosts. Furthermore, ports for less popular services are also closed less frequently on established hosts, with MongoDB and Microsoft SQL Server restricted on only 12% and 3% of hosts, respectively. This difference in port access changes highlights the fluid nature of newly-created hosts, which exhibit changes in security posture in the time after initial creation.

Overall, we find that, administrators neglect to revoke public access to sensitive services after site creation, and of those that do, many take hours to days to do so, providing attackers with ample opportunity to fingerprint and abuse these unprotected services. Attackers can trivially identify these sensitive services and compromise them through methods such as credential brute forcing or vulnerability exploitation. For this reason, it is of the utmost importance for administrators to ensure that access to these services is restricted prior to creating a TLS certificate.

TABLE I: Most common web server software used by sites in our dataset as determined by the HTTP `Server` header provided by each site. We find a large percentage of sites include not only the web server name, but also version number in the HTTP header. We identify a subset of these sites as being vulnerable to at least one CVE.

Server	Total	Total w/ Version	Vulnerable	Avg. CVSS
Nginx	104,813	27,522	5,954	6.0
Apache	98,072	23,837	18,750	5.1
Openresty	49,183	4,389	1,152	6.1
Litespeed	32,427	18	0	N/A
Microsoft	5,089	5,085	212	5.0
Other	183,363	32,265	863	6.8

3) Web Server Security: Common among the many available web server software packages today is the requirement for users to manually edit certain configuration options to ensure increased security posture. These include the restriction of certain file paths through access-control mechanisms, and tuning of the amount of information regarding the state of the web server exposed to the user through headers and web content [40], [41]. Below, we analyze four such configuration options and the propensity of administrators to neglect to properly tune each of these options prior to TLS certificate creation.

Server Version String Leakage

To assist in debugging, it is common for web server software to include by default an HTTP `Server` header indicating not only the name of web server handling the current request, but also the version number. The inclusion of such information can lead to serious security weaknesses as an attacker can easily map vulnerabilities to the specified software version to inflict harm against the web server. For this reason, it is common for administrators to remove/sanitize the `Server` header in all responses.

During our data collection period, we found 453,546 (82.73%) unique domains that sent an HTTP `Server` header in their responses, including 92,524 (16.88%) domains that also included the entire web server software version string. Of these, we observed that 62.57% changed their reported user agent after our first crawl, including those that completely removed the HTTP `Server` header from responses.

As leaking the entire version string of a web server through HTTP `Server` headers can lead to trivial exploitation by attackers, we sought to determine the number of these web servers that were vulnerable to known exploits. We queried the popular CVE database cvedetails.com for each version string in our dataset and recorded if that version is vulnerable to at least one exploit, along with the highest CVE score listed. In total, we found 26,887 domains in which the listed web server version was vulnerable to at least one CVE. Of these, the average CVE Score was 5.0 out of 10. Table I shows the breakdown of popular web server instances in our dataset. Table VI in the Appendix shows all the CVEs that we discovered affecting prematurely-public web applications.

Of these hosts, 9,857 (36.7%) either completely removed the `Server` header revealing the vulnerable web server software, or updated the software to a version that is not associated with a CVE before our last scan. This corresponds to 8,895 domains that removed the `Server` header and 962 that updated the software. Alarmingly, sites in our dataset were not quick to resolve this information leakage, with a median time-to-removal of 1 day. Beyond this window, attackers would miss 36.7% of vulnerable servers that could have been exploited.

To compare the difference in attack surface of new sites appearing on CT, to sites renewing their certificates, we conduct an experiment in which we attempt to analyze the server headers included in responses from established sites in our dataset. Specifically, long-lived sites that have had time to fully secure their infrastructure. To do this, we gathered the hosts we filtered from our dataset earlier that were found to be renewing their TLS certificate at the time our discovery (as opposed to obtaining a certificate for the first time). Moreover, to ensure we are analyzing only well-established sites, we further filtered this dataset by keeping only hosts with a primary domain in the Tranco [42] Top 100K. This left us with a dataset of 1,249,061 total crawls of 264,929 domains.

Of these hosts, we find 176,970 (66.8%) return an HTTP `Server` header to requests. Contrary to newly-created sites however, only 22,397 (8.45%) of these server headers contain a version number. This is not surprising as established sites have had the time to fully harden their security posture, including improving operational security with the removal of version string leakage.

This does not mean however that all sites have removed this vulnerability. In this dataset, we find 7,370 (2.8%) sites that presented version strings for web servers vulnerable to at least one public CVE. The average score of these CVEs is in line with our new-site dataset at 6.0 out of 10. Of these hosts, 2,249 were either removed completely, or changed in such a way that the eventual server version was no longer vulnerable. These changes occurred with a median-time-to-removal of 24 days. Given the small relative number of sites exhibiting this behavior in the "established-sites" group (2.2K out of 177K) we interpet this phenomenon as the result of administrators updating their software only when they are made aware of issues, as opposed to the typical online hardening we observed for websites that came online for the first time.

These results clearly demonstrate the tendency of administrators to create TLS certificates in the middle of their website configuration workflows. It is not likely that these websites were intended to be fully deployed on outdated and vulnerable web server software. Instead, these administrators used the outdated software to install/configure the website before updating software prior to deployment. However, due to a lack of knowledge of the public nature of CT, TLS certificates were created for these domains before the underlying web servers were properly updated and secured.

Directory Access

An important aspect of web server setup is the presence of access control, restricting the ability of users to request certain

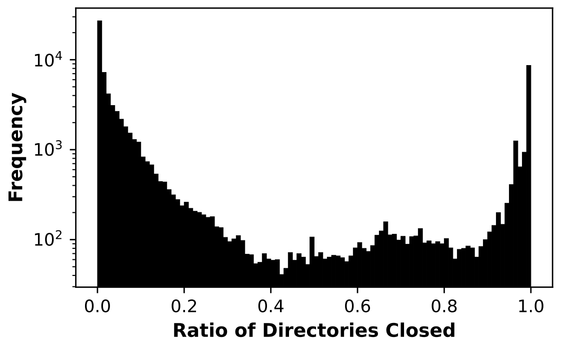

Fig. 8: Distribution of the ratio of the number of directories in which each domain revoked access after certificate creation.

resources. An example of this is the `.htaccess` file of the Apache web server software [43], which allows for access control rules to be applied on a per-directory basis. Failure to properly configure such restrictions can allow attackers to interact with sensitive parts of a web application, such as administrator panels or login pages.

In total, we found 172,087 domains that have a change in the number of accessible directories during our data collection period, with 76,735 (44.6%) domains decreasing the number of available directories. Figure 8 shows the distribution of the ratio of total directories removed in our data collection period. We observe a bimodal distribution, where a large percentage of domains remove access to only a handful of directories, whereas a second cluster of domains remove access to almost all originally open directories. These changes tend to occur rapidly, with the majority removing access within an hour of our first crawl.

Directory Indexes

Another common debugging feature of web server software is the listing of file system directories when receiving a GET request for that particular directory. In its intended use, this feature allows for easy identification of files accessible through a web server. However, it can also provide attackers with knowledge of the existence of potentially sensitive files, greatly increasing their intelligence prior to launching an attack. To determine a web server's current support for this behavior, we use a series of regular expression searches on the responses to the same requests we transmitted to determine a web server's directory access control settings, corresponding to the structure of directory indexing web pages of the most popular web server software packages.

In total, we find 26,993 domains that list the contents of at least one directory during our crawls. Of these, 9,527 (35.3%) domains removed the support for directory indexing, with this change occurring a median of 1 day after our first successful crawl. This finding, along with the domains that lack access-control measures on web server directories show how attackers utilizing a system such as MAKO can quickly map a web server's file system, and identify sensitive files.

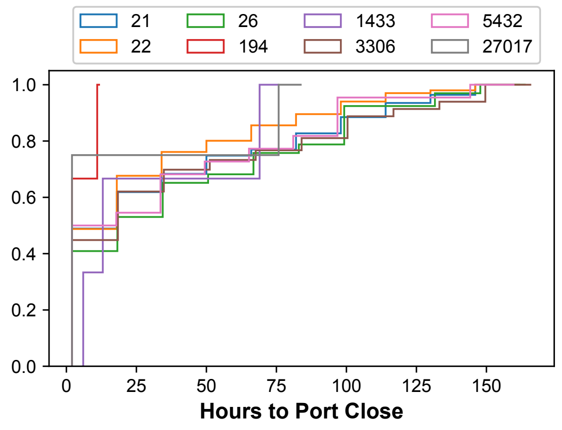

Fig. 9: Distribution of times in which hosts closed sensitive network ports during our fine-grained analysis of sites every two hours for a week.

4) Web Content: We found that 69.3% of all domains in our dataset return HTML content at some point during our crawls, with 50.1% returning HTML content on our first crawl immediately after appearing on CT. To determine the types of content served at various crawls, we extract the HTML `title` tag of each page and cluster based on this value. We then label these clusters based on the content. In Section VI, we discuss specific categories of web pages that indicate sensitive content is being publicly shared by the web server.

V. FINE-GRAINED SECURITY AUDITING

In Section IV, we discovered that websites do indeed improve their security posture in the hours to days following certificate creation. This vulnerability window could allow quick attackers to exploit weaknesses in a site before they are patched. However, in order for MAKO to be able to scan a large number of sites and produce generalizable findings, we chose a coarse-grained recrawling interval, visiting sites one hour, one day, and one week after original discovery. While this decision allowed us to observe online changes to the security posture of sites, it did not allows us to identify with precision when exactly administrators harden their web applications.

Thus, we conduct a small-scale supplementary experiment, with a more fine-grained recrawling interval. Specifically, we utilize the same methodology described in Section III and randomly sample 2,457 domains from CT logs over a 24 hour period. MAKO crawled each domain immediately after first discovery, as well as each every two hours for the subsequent week, resulting in 84 additional crawls. As this recrawling interval increases the number of probes directed towards each target site from MAKO, we limit our measurements to only open/closed network ports.

Figure 9 shows the distribution of port closures of a subset of sensitive ports during our weeklong supplementary data collection period. Overall, we observe varying behavior of hosts closing sensitive ports, with over half of all analyzed sites closing these ports over a day after first discovery. Moreover, we find that hosts are more likely to close ports for sensitive data providing services, such as 1433 (Microsoft SQL server), far quicker than other ports, with nearly 75% of all hosts closing that port within two hours of our first crawl. The varying behavior observed among hosts in this dataset is due to both our smaller sample size, as well as the thousands of different stakeholders (with different practices) involved in launching new web applications online.

VI. Case Studies

A. Public WordPress Installation Forms

The Internet would not be the same without the open-source web application WordPress [44]. WordPress is a content-management system used by 43% of *all* websites on the web [45]. It is therefore not surprising that it is a highly sought-after target for attackers [46], [47], who utilize the standardized nature of this massively-deployed web application to identify and exploit vulnerable installations, including knowledge on the installation process and locations of version strings within the web page content.

To appeal to less technically-savvy users, WordPress confines its installation process to a web-based GUI, making it easy for users to input configuration options such as the address of their MySQL database and the desired site administrator credentials. This significantly lowers the barrier to entry for users that create their own WordPress sites, likely contributing to its overwhelming market share. However, this choice leads to an obvious vulnerability: if proper care is not taken, the installation process can be hijacked by a quick attacker that visits the site at the time of installation. This attack was first demonstrated in 2017, when over four thousand web application installer pages were discovered by comparing the HTML of web pages returned from sites appearing on CT to those of known installers [48].

Using the HTML title tag clustering technique from Section IV-A4, we identify 211,880 sites in our dataset that indicate their use of WordPress through the inclusion of the string “WordPress” in the HTML title. We note that this is likely a lower-bound of WordPress sites in our dataset, as some administrators may wish to remove indications of a site’s use of a particular web application.

Alarmingly, we found 873 domains that were actively installing and configuring WordPress at the time of our first crawl, with these sensitive pages only available for 12 hours on average, before installation is completed. During this time period, the sensitive forms used to completely configure the host site were publicly accessible to any visitor in the hours after certificate creation, opening a severe vulnerability window that will not exist for a long time.

We note that while these WordPress installations could have been abused at the time of their discovery to run arbitrary server-side code, this is entirely the result of operational errors on the part of these website administrators, rather than a vulnerability in the source code of the WordPress application itself. By the time we analyzed our data to discover this vulnerability, these sites were no longer vulnerable, hence there was no reason to contact them.

B. SQL Error Messages

As we showed in Section IV-A4, many sites return server error messages to GET requests on our first crawl, before these errors disappear in the subsequent crawls. The implication is that the site is still in the process of being deployed, with error messages indicating incomplete configuration. Although many of these errors messages are inconspicuous, we identify a set of SQL-based error messages that can reveal sensitive information regarding the backend database. Again, we utilize HTML title tag clustering to identify these pages, by first searching for all titles containing the string “SQL” and then manually filtering the resulting list to only keep those related to error messages.

In total, we found 69 domains that returned a SQL error message to the GET requests during our first crawl. This can be a highly damaging leakage of information, as these error messages reveal vital data regarding the structure of the application’s database, as well as information regarding the operation of the application itself. For instance, 30 error messages include the name of the database user that performed the query. Additionally, these errors include information regarding how authentication is performed by the database server. Of the messages that included the database username, 14 state that the database uses password authentication. With this information, along with public access to the database server port (seen in Section IV-A2), an attacker can attempt to brute force access into the database with a strong starting position.

In addition to authentication information, SQL error messages can reveal hidden vulnerabilities in the logic of the application through the leakage of the raw SQL query strings used to interact with the database. In total, 46 domains in our dataset returned error messages including a raw SQL query string. The most egregious example in our dataset is one site that was running a Laravel PHP web application that had a connection error to the SQL server. This site returned to the user a full stack trace revealing a severe vulnerability within the application logic where a user’s session ID is retrieved from the database using only the HTTP User-Agent, shown in Code Listing 1 of the Appendix. We contacted the owner of this site and disclosed our findings to ensure this vulnerability is fixed before it can be exploited by attackers. Examples such as this provide attackers with substantial information on how to exploit a particular web application. It is important for administrators to ensure that errors are handled properly before creating a TLS certificate for their site, to ensure that sensitive information is not leaked.

C. Vulnerable OpenSSH Servers

In Section IV-A2, we demonstrated the propensity of web hosts to leave sensitive network services open to the public upon certificate creation, before revoking access at a later time. One of the most vital, yet sensitive of these services is SSH, located on port 22 by default. By connecting to an SSH server on a remote host, clients can run commands and manipulate files as if they were local to the machine. Like many other network protocols, communication with an SSH server begins with the server providing the client with a banner displaying the current version of the SSH protocol supported, and the underlying SSH server software identifier. While there exists a number of SSH server software packages available, the most popular of these is OpenSSH [49], integrated into many of the most popular operating systems in use today [50].

Using the banners recorded during our port scan probes in Section IV-A2, we identify all hosts utilizing OpenSSH on port 22 as well as all versions of OpenSSH installed during each crawl. In total, we found 155,840 hosts that present an OpenSSH server to the Internet. Alarmingly, 151,663 (97.3%) of these hosts utilize a version of OpenSSH that is outdated and vulnerable to at least one known CVE, with an average CVE Score of 5.0. In Section VII, we further investigate these vulnerable servers and find approximately 20% originate from a single Autonomous System. Of all hosts that are observed running a vulnerable version of OpenSSH, 16,418 (10.8%) resolve this vulnerability by either updating the software, or removing access to port 22 entirely at a later crawl.

VII. Discussion

A. Key Takeaways

Prematurely-public Web Applications: Our results show that there is a substantial population of websites that, due to the CT system, announce their existence to the world before they intended, opening the door for attackers to exploit vulnerabilities that will only exist for a short period of time. In total, we observed 200,421 domains that improved their security posture in the hours to days after the creation of their TLS certificate. For many domains, severe vulnerabilities existed in their web infrastructure that could allow attackers to gain a foothold before all security measures were applied. Overall, our findings demonstrate the importance for web administrators to understand the implications of CT and to ensure that security-related measures are applied prior to creating a TLS certificate.

Certificate Transparency Volume: Within a 10 week study, we were able to find tens of thousands of sites vulnerable to many of the security weaknesses we measured, despite the use of a 1% sampling rate of domains on CT. The true number of sites that improve their security posture after creating a TLS certificate is much larger. As is the case with any studied security vulnerability, the immense scale of the Internet provides attackers with hundreds of thousands to millions of potential targets with weakened security postures immediately after TLS certificate creation.

CT Education: During the course of our data collection period, we received a number of emails from concerned web administrators inquiring how we were able to identify their newly-created domains as quickly as we did. For each such request, we provided thorough information regarding the implications of creating a TLS certificate and the potential impacts of announcing new domains prior to fully securing them. These interactions demonstrate the current dearth in knowledge of the CT system and the false sense of security currently assumed by administrators from the perceived secrecy of newly-created domains and subdomains. It is our hope that our study can help further educate the community on the security implications of CT and best security practices when configuring a new web endpoint.

Privacy-preserving CT: In light of the previously-described requirement of administrators to be cognizant of the public nature of TLS certificate creation, researchers have begun to question the breadth of information required to be logged by CT for its security guarantees. Proposals for a replacement of the CT system include the utilization of cryptographic means to verify a certificate's validity, rather than logging of the raw metadata [51], [52]. Removing the raw metadata would take the onus off of administrators to ensure certificates are only created when the underlying web server is at its ultimate security posture. We encourage the research community to continue its efforts in developing a privacy-preserving alternative to CT.

Hosting Provider Vulnerabilities: While many of the security indicators we study in this work can typically be attributed to the actions of the website admin, such as the improper configuration of access control methods, we note that some can be a result of the improper configuration of pre-made virtual machine templates supported by particular web hosting providers. For instance, if a hosting provider allows customers to purchase pre-configured virtual machines containing all web and database software necessary to launch a web application, but does not properly maintain such an image to ensure all software is up-to-date, any vulnerability present in the software of that image will be deployed en masse by that hosting provider's users.

Figure 10 shows the ratio of all hosts that demonstrate weak security posture in one of the studied categories, for the top 20 most common Autonomous Systems in our dataset. By comparing an Autonomous System's overall share of sites in our dataset (listed in the "Total Sites" column), to its share of all sites exhibiting a weakened security posture in a particular category, we can see a number of hosting providers that disproportionately appear. For example, while the hosting provider *UnifiedLayer* makes up only around 5% of hosts in our dataset, it is responsible for almost 20% of all hosts that provide access to a vulnerable OpenSSH server. We therefore note the possibility of temporal vulnerabilities due to

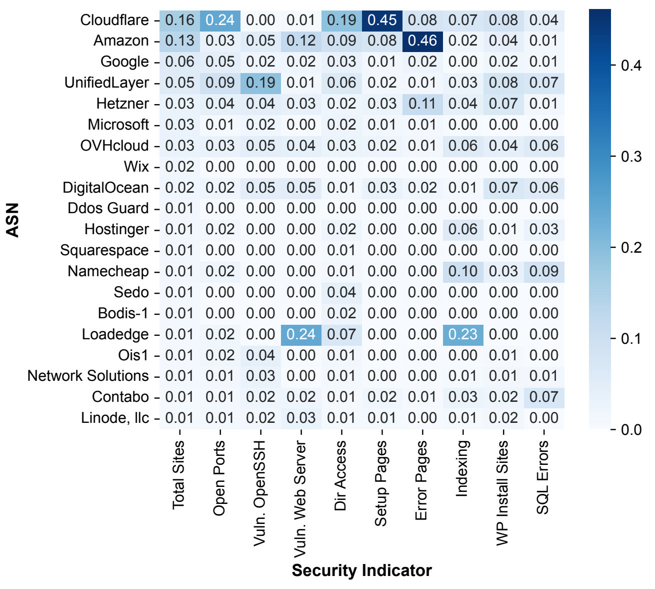

Fig. 10: *Percentage of unique domains that had a weakened security posture in each of the studied categories for the top 20 most common ASNs in our dataset.*

outdated pre-made virtual machines and leave a more detailed exploration of this phenomenon for future work

B. Limitations & Future Work

Our analysis should be considered alongside certain limitations. Due to the significant number of new TLS certificates created every day, we used a 1% sampling rate on domains sourced from CT logs. Therefore, MAKO was only able to analyze a small fraction of all possible sites during our data collection period. Although we apply some filtering logic, as described in Section III, the majority of our filtering steps were applied after the conclusion of our data collection period. This is mostly due to the time required to query for post-filtering information, such as historical CT logs. Future uses of a system such as MAKO should attempt to conduct most filtering in real-time to increase the number of relevant domains in the final data set. For instance, identifying newly-created domains rather than certificate renewals of existing domains in real-time would allow for greater worker capacity.

Additionally, our study focused entirely on server-side vulnerabilities present in sites at the time of certificate creation. We made the choice to focus on these security posture weaknesses as opposed to factors primarily affecting end-users because the sites listed on CT are new, and thus do not yet have active users. We encourage future work to study the changes in user-side security posture of sites over time after TLS certificate creation. This can include factors such as TLS and HTTP versions supported by the web server, as well as the inclusion of security-related headers and web content such as Content-Security-Policy and Subresource Integrity tags.

C. Ethical Considerations

During our study, we took care to ensure MAKO's probes were not harmful or disruptive to the sites we analyzed. Each site visited by MAKO received a limited number of network requests to publicly-accessible endpoints. At no point did MAKO probe for vulnerabilities or use excessive web server resources. Moreover, MAKO never transmitted a request to update web server state, such as an HTTP POST or DELETE request. We performed a limited scan on 20 ports, sending a single TCP packet to each port. This is in line with prior work, such as ZGrab, which scanned 16 ports on each host [29].

Although expanding MAKO's interaction with each host past the probes described in this paper could result in interesting security posture findings, we limit MAKO for ethical reasons. For instance, by completing full SSH interactions with each host, MAKO could discover the adoption of IDS systems (such as fail2ban [53]), weak/default password use, and transition from password-based to key-based authentication. Notably, although we limited MAKO in this regard, attackers are not ethically bound, and would likely identify additional vulnerabilities that can be used to exploit hosts.

Finally, during our data collection period, we promptly addressed all opt-out requests, propagating updated blocklists to all MAKO instances. We also immediately responded to all requests for collected data to the administrators of the scanned sites and removed from our dataset all data pertaining to the specified sites when requested to do so.

VIII. RELATED WORK

Certificate Transparency

Certificate Transparency has undoubtedly improved the security of the modern web by increasing the trust the community has of the certificates used by websites. However, its introduction has also led to some unintended security and privacy consequences that have been explored in prior work. A number of studies have been conducted to understand the fundamental nature of CT, including properties of the system itself as well as actors involved in the ecosystem [54]–[58].

Prior work has also explored the privacy impacts of CT. A major focus of this work has been the impact of the logging of potentially-sensitive information regarding certificate registrations [51], [52], [59]–[62]. This includes its use as a vector for domain discovery [63]–[65]. Kondracki et al. explored the ecosystem of bots utilizing CT for target acquisition by creating a CT domain honeypot system, called CTPOT. The system works by periodically creating certificates for pseudo-random subdomains and measuring the resulting bot activity directed towards those domains from a variety of network vantage points. The authors found that bot activity towards domains appearing on CT begins in as little as 12 seconds after certificate creation, many of which produce malicious payloads [6]. In this paper, we build upon these findings to measure the security posture of hosts as they appear on CT. We find that the bots utilizing CT to acquire scanning targets can expect to find thousands of vulnerable hosts, many of which will no longer be available to attackers in the hours after first appearance on CT.

Web Host Security Scanning

Today, analysis on the security posture of web hosts is a mature research field, spanning all levels of the network stack, and targeting hosts from various sources. Security of web server software has been examined from the perspective of the attack surfaces in terms of open ports [66], [67], as well as the configuration of vital mechanisms such as TLS and DNS [68]–[70], and security measures within the content produced [7], [8]. Some work has also taken a holistic approach to security posture analysis, analyzing website security measures focused in particular world regions [9], [10].

The standard in IPv4 web host scanning has come from the work of Durumeric et al. with ZMap [71]. ZMap improved upon prior Internet scanning tools by significantly decreasing the time required to scan the entire IPv4 address space. This led to follow-up work by Durumeric et al. in which the Censys search engine was created, which allows researchers and security professionals to query up-to-date information on hosts across the Internet [29].

This paper takes inspiration from the work of Bock, who demonstrated how attackers could utilize CT logs to hijack the installation of popular web applications [48]. The author scanned web pages from hosts found on CT logs and compared the resulting web pages to known web application installers to determine if they could be hijacked by a quick attacker. Through this attack methodology, the author discovered over four thousand sites running popular web applications such as WordPress, Joomla, and Owncloud that were being installed at the time of first visit. Our work builds upon that of Böck by expanding this analysis to multiple layers of the network stack, including scans of popular network ports, analysis of vulnerable web server software, and web server access control measures. Additionally, MAKO's revisiting of sites in the days following first appearance on CT allows us to quantify the window of vulnerability created by administrators configuring security measures after certificate creation.

In concurrent work, Pletinckx et al. study the implications of CT logs on unwanted scanning traffic [72]. In addition to measuring the volume of traffic directed at domains appearing on CT logs, the authors also utilize CT logs to compare the security posture of newly-created websites vs. established websites with expired certificates. Unlike our work, the authors study a limited set of sampled domains from a single day, recrawling each site a handful of times during the following week. Moreover, the authors measure the security posture of a site utilizing only the HTTP headers and index HTML code of each site, thereby significantly underestimating the vulnerability of these CT-discovered sites.

In our paper, we quantify the dangers of premature CT announcements, going far beyond all prior work both in terms of scale as well as analysis. Our longitudinal, fixed-interval scans in the hours to days after their certificate issuance allowed us to track the hardening of sites in near realtime. This is not the case for studies that utilize shorter scanning durations, such as [72], leading to underestimating the number of sites that become more secure post-certificate-creation.

Moreover, the scans conducted by MAKO allow us to analyze the security posture of online hosts at a greater depth than prior work. We identify security vulnerabilities in sites appearing on CT from multiple levels of the network stack, ranging from misconfigured network firewalls, to incorrectly-public web application installation forms. Our results demonstrate that the web administration community is largely unaware of the negative consequences of the CT ecosystem, leading to these wide-ranging misconfigurations. We therefore hope that this work will help educate website administrators of the dangers of premature TLS certificate creation, and will lead to the proliferation of better development workflows to safeguard newly-created websites from attack.

IX. CONCLUSION

In this paper we presented MAKO, a web-infrastructure security auditing system that ingests CT logs to discover newly-created domains and quantify their security posture. In total, we audited the security measures of 548,238 unique web hosts that did not exist prior to our first discovery. By visiting each host multiple times over the days following its original certificate creation, we were able to measure the delta in security posture of this period, ultimately discovering 200,421 (36.5%) sites that begin in a vulnerable state before eventually resolving these vulnerabilities. These vulnerabilities range from the network-level, with 124,307 domains closing at least one sensitive network port in the hours after certificate creation, to the content produced by web servers with 8,895 domains advertising the use of vulnerable web-server software, before either hiding that information or updating the web server to a non-vulnerable version.

Our results demonstrate the current lack of understanding among administrators on the security implications of obtaining TLS certificates. While, in the past, it could be assumed a new endpoint was hidden until purposely advertised, the introduction of CT has moved that time to the instant a certificate is created for an endpoint. We discovered that attackers that abuse this temporal discrepancy can expect to find between 15-25% more vulnerable hosts than by scanning only established sites. Moreover, we find this window of vulnerability can be open for multiple days after certificate creation, with thousands of sites that can be exploited before vulnerabilities are addressed. Overall, we hope our findings help to educate the community on the implications of creating a TLS certificate, and push future research for more secure web workflows, as well as privacy-preserving certificate announcements.

Acknowledgments: We thank the anonymous reviewers and shepherd for their help. This work was supported by the Office of Naval Research (ONR) under grant N00014-24-1-2193, the Army Research Office (ARO) under grant W911NF-24-1-0051 as well as by the National Science Foundation (NSF) under grants CNS-2211575 and CNS-1941617.

Availability: To assist in furthering the understanding of the kinds of vulnerabilities present on web hosts at the time of certificate creation, we make anonymized portions of our dataset available along with the source code of MAKO [73]

References

[1] "Usage statistics of default protocol https for websites," https://w3techs.com/technologies/details/ce-httpsdefault.

[2] "Certbot," https://certbot.eff.org, 2023.

[3] "Tls/ssl automation in digicert certcentral," https://www.digicert.com/solutions/security-solutions-for-automation, 2023.

[4] "Caddy server," https://caddyserver.com, 2023.

[5] "Certificate transparency," https://certificate.transparency.dev, 2023.

[6] B. Kondracki, J. So, and N. Nikiforakis, "Uninvited guests: Analyzing the identity and behavior of certificate transparency bots," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 53–70.

[7] 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.

[8] Z. Kang, S. Li, and Y. Cao, "Probe the proto: Measuring client-side prototype pollution vulnerabilities of one million real-world websites," in *Network and Distributed System Security Symposium (NDSS)*, 2022.

[9] P. Chen, J. Visschers, C. Verstraete, L. Paoli, C. Huygens, L. Desmet, and W. Joosen, "The relationship between the cost of cybercrime and web security posture: a case study on belgian companies," in *Proceedings of the 11th European Conference on Software Architecture: Companion Proceedings*, 2017, pp. 115–120.

[10] J. Mtsweni, "Analyzing the security posture of south african websites," in *Information Security for South Africa (ISSA)*. IEEE, 2015, pp. 1–8.

[11] "Comodo incident," https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html, 2023.

[12] "Turktrust incident," https://blog.mozilla.org/security/2013/01/03/revoking-trust-in-two-turktrust-certficates/, 2023.

[13] "Anssi incident," https://www.mozilla.org/en-US/security/advisories/mfsa2013-117/, 2023.

[14] "Diginotar hack," https://slate.com/technology/2016/12/how-the-2011-hack-of-diginotar-changed-the-internets-infrastructure.html, 2023.

[15] "Chrome certificate transparency policies," https://chromium.googlesource.com/chromium/src/+/refs/heads/main/net/docs/certificate-transparency.md, 2023.

[16] "Apple's certificate transparency policy," https://support.apple.com/en-gb/HT205280, 2023.

[17] "Opera browser - misissued certificates," https://blogs.opera.com/security/2015/10/misissued-certificates/, 2023.

[18] "Certificate transparency - acm queue," https://queue.acm.org/detail.cfm?id=2668154, 2023.

[19] "Facebook certificate transparency api," https://developers.facebook.com/docs/certificate-transparency-api/, 2023.

[20] "Google certificate transparency sdk," https://github.com/google/certificate-transparency, 2023.

[21] "Certstream," https://certstream.calidog.io, 2023.

[22] "Https encryption on the web," https://transparencyreport.google.com/https/overview?hl=en, 2023.

[23] "Nmap," https://nmap.org, 2023.

[24] "Zmap," https://zmap.io, 2023.

[25] "Seleium," https://www.selenium.dev, 2023.

[26] "nitko," https://github.com/Tib3rius/nitko, 2023.

[27] "wpscan," https://wpscan.com/wordpress-security-scanner, 2023.

[28] "w3af," http://w3af.org, 2023.

[29] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, "A search engine backed by internet-wide scanning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 542–553.

[30] "Shodan," https://www.shodan.io, 2023.

[31] "Metasploit," https://www.metasploit.com, 2023.

[32] "Github api," https://docs.github.com/en/rest, 2023.

[33] "How to prevent directory listing of your website with .htaccess," https://www.thesitewizard.com/apache/prevent-directory-listing-htaccess.shtml, 2023.

[34] "Report: State of the web," https://httparchive.org/reports/state-of-the-web#reqTotal, 2023.

[35] "crt.sh," https://crt.sh, 2023.

[36] "Service subdomains explanation," https://documentation.cpanel.net/display/CKB/Service+Subdomains+Explanation?_ga=2.117030149.2095746051.1669735322-882536857.1669735322, 2023.

[37] "cpanel: Autoconfig and autodiscover," https://docs.cpanel.net/knowledge-base/email/autoconfig-and-autodiscover/, 2023.

[38] "Microsoft sql servers hacked in targetcompany ransomware attacks," https://www.bleepingcomputer.com/news/security/microsoft-sql-servers-hacked-in-targetcompany-ransomware-attacks/, 2023.

[39] "22,900 mongodb databases affected in ransomware attack," https://www.darkreading.com/cloud/22-900-mongodb-databases-affected-in-ransomware-attack, 2023.

[40] T. Mobily, *Hardening Apache*. Apress, 2004.

[41] M. D. Bauer, *Linux server security*. " O'Reilly Media, Inc.", 2005.

[42] "Tranco," https://tranco-list.eu, 2023.

[43] "Apache http server project," https://httpd.apache.org, 2023.

[44] "Wordpress," https://wordpress.org, 2022.

[45] "Usage statistics and market share of wordpress," https://w3techs.com/technologies/details/cm-wordpress, 2022.

[46] "1.6 million wordpress sites hit with 13.7 million attacks in 36 hours from 16,000 ips," https://www.wordfence.com/blog/2021/12/massive-wordpress-attack-campaign/, 2021.

[47] R. P. Kasturi, J. Fuller, Y. Sun, O. Chabklo, A. Rodriguez, J. Park, and B. Saltaformaggio, "Mistrust plugins you must: A large-scale study of malicious plugins in wordpress marketplaces," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 10–12.

[48] H. Böck, "Abusing certificate transparency or how to hack web applications before installation," in *DEFCON 25*, 2017.

[49] "Openssh," https://www.openssh.com, 2023.

[50] "Openssh users," https://www.openssh.com/users.html, 2023.

[51] S. Eskandarian, E. Messeri, J. Bonneau, and D. Boneh, "Certificate transparency with privacy," *arXiv preprint arXiv:1703.02209*, 2017.

[52] H. Kwon, S. Lee, M. Kim, C. Hahn, and J. Hur, "Certificate transparency with enhanced privacy," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[53] "Fail2ban," https://www.fail2ban.org/wiki/index.php/Main_Page, 2023.

[54] J. Gustafsson, G. Overier, M. Arlitt, and N. Carlsson, "A first look at the ct landscape: Certificate transparency logs in practice," in *International Conference on Passive and Active Network Measurement*. Springer, 2017, pp. 87–99.

[55] B. Li, J. Lin, F. Li, Q. Wang, Q. Li, J. Jing, and C. Wang, "Certificate transparency in the wild: Exploring the reliability of monitors," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2505–2520.

[56] C. Nykvist, L. Sjöström, J. Gustafsson, and N. Carlsson, "Server-side adoption of certificate transparency," in *International Conference on Passive and Active Network Measurement*. Springer, 2018, pp. 186–199.

[57] E. Stark, R. Sleevi, R. Muminovic, D. O'Brien, E. Messeri, A. P. Felt, B. McMillion, and P. Tabriz, "Does certificate transparency break the web? measuring adoption and error rate," in *IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 211–226.

[58] B. Dowling, F. Günther, U. Herath, and D. Stebila, "Secure logging schemes and certificate transparency," in *European Symposium on Research in Computer Security*. Springer, 2016, pp. 140–158.

[59] D. Kales, O. Omolola, and S. Ramacher, "Revisiting user privacy for certificate transparency," in *IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 432–447.

[60] R. Roberts and D. 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*, 2019, pp. 87–92.

[61] Q. Scheitle, O. Gasser, T. Nolte, J. Amann, L. Brent, G. Carle, R. Holz, T. C. Schmidt, and M. Wählisch, "The rise of certificate transparency and its implications on the internet ecosystem," in *Proceedings of the Internet Measurement Conference*, 2018, pp. 343–349.

[62] T.-D. Nguyen, "Measuring the impact of certificate transparency on scanning traffic," 2022.

[63] B. Kondracki, B. A. Azad, O. Starov, and N. Nikiforakis, "Catching transparent phish: Analyzing and detecting mitm phishing toolkits," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 36–50.

[64] A. Drichel, V. Drury, J. von Brandt, and U. 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*, 2021, pp. 1–12.

[65] 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 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 569–586.

[66] Y. J. Jia, Q. A. Chen, Y. Lin, C. Kong, and Z. M. Mao, “Open doors for bob and mallory: Open port usage in android apps and security implications,” in *IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2017, pp. 190–203.

[67] M. A. Sulaiman and S. Zhioua, “Attacking tor through unpopular ports,” in *IEEE 33rd International Conference on Distributed Computing Systems Workshops*. IEEE, 2013, pp. 33–38.

[68] C. Simoiu, W. Nguyen, and Z. Durumeric, “An empirical analysis of https configuration security,” *arXiv preprint arXiv:2111.00703*, 2021.

[69] D. Springall, Z. Durumeric, and J. A. Halderman, “Measuring the security harm of tls crypto shortcuts,” in *Proceedings of the Internet Measurement Conference*, 2016, pp. 33–47.

[70] C. Wang, K. Shen, M. Guo, Y. Zhao, M. Zhang, J. Chen, B. Liu, X. Zheng, H. Duan, Y. Lin *et al.*, “A large-scale and longitudinal measurement study of {DKIM} deployment,” in *31st USENIX Security Symposium (USENIX Security)*, 2022, pp. 1185–1201.

[71] Z. Durumeric, E. Wustrow, and J. A. Halderman, “{ZMap}: Fast internet-wide scanning and its security applications,” in *22nd USENIX Security Symposium (USENIX Security)*, 2013, pp. 605–620.

[72] S. Pletinckx, T.-D. Nguyen, T. Fiebig, C. Kruegel, and G. Vigna, “Certifiably vulnerable: Using certificate transparency logs for target reconnaissance,” in *8th IEEE European Symposium on Security and Privacy*. IEEE, 2023.

[73] “Mako project website,” https://pragseclab.github.io/mako.

Appendix

We include the following tables as supplemental material to the paper.

TABLE II: *List of ports scanned by* MAKO *along with the most common service that listens on each port.*

Port	Service
21	FTP
22	SSH
23	Telnet
25	SMTP
26	SMTP
53	DNS
80	HTTP
110	POP3
143	IMAP
194	IRC
443	HTTPS
993	IMAP-SSL
995	POP3-SSL
1433	MS SQL Server
3306	MySQL
5432	Postgresql
8080	HTTP
8443	HTTPS
8888	HTTP
27017	MongoDB

TABLE III: *List of regular expressions used to identify directory listing webpages.*

Regex String
Index of /
Go up</span
Directory Listing For \\[
To Parent Directory

TABLE IV: *List of directories requested from each web host.*

tmpl	user	icons
public	cache	blog
images	database	underscore
views	plugins	mssql
web	config	mysqli
test	en-gb	postgre
html	categories	oci8
css	fields	odbc
helpers	fonts	sqlite
app	themes	pdo
js	dist	cubrid
src	application	includes
models	1	akismet
examples	tables	languages
demo	langs	compat
img	site	_site
templates	mysql	api
docs	category	php
static	logs	javascript
language	errors	lib
controllers	client	misc
libraries	example	documentation
drivers	sql	javascript
www	wp-content	session
assets	network	search
doc	form	sqlite3
tests	description	subdrivers
forms	hooks	samples
admin	classes	xml
system	cli	ibase
webapp	third_party	about
core	browser	media
english	sqlsrv	wp-includes
default	cache	wp-content
		backup

TABLE V: *Top 10 most common HTTP Server header transitions.*

First Header	Modified Header	%
N/A	Cloudflare	19,333 (14.01%)
Apache	N/A	18,283 (13.25%)
Cloudflare	N/A	18,273 (13.24%)
Nginx	N/A	15,592 (11.30%)
N/A	Nginx	11,628 (8.43%)
N/A	Apache	8,420 (6.10%)
Litespeed	N/A	8,324 (6.03%)
Openresty	N/A	6,557 (4.75%)
N/A	Openresty	5,989 (4.34%)
N/A	Litespeed	4,012 (2.91%)

```
use Share\Models\Session;
use Illuminate\Http\Request;

class ShareAuthMiddleware
{
    public function handle(Request $request, Closure $next)
    {
        if (! auth()->user()) {
            $user_agent = $request->header('User-Agent');
            $log = Session::where('user_agent', $user_agent)
                 ->whereNotNull('user_id')->first();
            if ($log && $log->user_id) {
                $domain = (string) \Str::of(url()->previous()
                     )->after('shop.')->before('.');
                if ($domain !== domain()) {
                    auth()->loginUsingId($log->user_id);
                }
            }
        }
        return $next($request);
    }
}
```

Listing 1: *Example of application logic error discovered by* MAKO *through SQL error message.*

TABLE VI: *Full list of CVEs discovered from HTTP and OpenSSH server version strings in our dataset.*

Akka-http	**Werkzeug**
CVE-2021-42697	CVE-2019-14806
AOLserver	CVE-2020-28724
CVE-2009-4494	CVE-2022-29361
Apache	**GoAhead-Webs**
CVE-2002-0392	CVE-2019-16645
CVE-2002-0661	**gunicorn**
CVE-2002-0843	CVE-2018-1000164
CVE-2004-0751	**httpd**
CVE-2005-2088	CVE-2002-1592
CVE-2006-3747	**Microsoft-IIS**
CVE-2011-3368	CVE-2005-2089
CVE-2013-6438	CVE-2008-0074
CVE-2015-3183	CVE-2010-1899
CVE-2016-0736	**Kong**
CVE-2017-3167	CVE-2021-27306
CVE-2017-3169	**Lighttpd**
CVE-2017-7679	CVE-2014-2323
CVE-2017-9789	CVE-2019-11072
CVE-2017-9798	**LMS**
CVE-2018-1303	CVE-2021-25200
CVE-2018-17189	**LWS**
CVE-2019-0196	CVE-2008-1042
CVE-2019-10081	**mini_httpd**
CVE-2019-17567	CVE-2009-4490
CVE-2021-26690	**MiniServ**
CVE-2021-31618	CVE-2004-1468
CVE-2021-36160	CVE-2018-8712
CVE-2021-42013	**NCSA**
CVE-2022-22719	CVE-1999-0236
CVE-2022-28614	**Nexus**
CVE-2022-36760	CVE-2020-13933
APISIX	**OpenCMS**
CVE-2022-29266	CVE-2019-11819
Apache Traffic Server	CVE-2021-3312
CVE-2021-35474	**Resin**
Boa	CVE-2008-2462
CVE-2017-9833	**Squid**
CVE-2018-21027	CVE-2016-10003
Nginx	**SRP**
CVE-2013-4547	CVE-2014-3512
CVE-2016-0746	**Sun-One**
CVE-2017-20005	CVE-2004-0826
CVE-2018-16843	**TwistedWeb**
CVE-2018-16844	CVE-2020-10108
CVE-2019-20372	**Uc-httpd**
Jetty	CVE-2018-10088
CVE-2018-12536	**uhttpd**
CVE-2018-12545	CVE-2019-19945
CVE-2021-28169	**Virata-EmWeb**
CVE-2021-34429	CVE-2006-0248
CVE-2022-2191	**WildFly**
Openresty	CVE-2020-25689
CVE-2018-9230	**Zope**
CVE-2020-11724	CVE-2011-3587
Synapse	**OpenSSH**
CVE-2022-31052	CVE-2006-5051
UPnP	CVE-2015-5600
CVE-2019-16903	CVE-2016-6515
	CVE-2021-41617