Certifiably Vulnerable: Using Certificate Transparency Logs for Target Reconnaissance

Stijn Pletinckx
University of California, Santa Barbara
Santa Barbara, USA
stijn@ucsb.edu

Thanh-Dat Nguyen
Delft University of Technology
Delft, the Netherlands
tdat.m.nguyen@gmail.com

Tobias Fiebig
Max Planck Institute for Informatics
Saarbrücken, Germany
tfiebig@mpi-inf.mpg.de

Christopher Kruegel
University of California, Santa Barbara
Santa Barbara, USA
chris@cs.ucsb.edu

Giovanni Vigna
University of California, Santa Barbara
Santa Barbara, USA
vigna@ucsb.edu

***Abstract*—The Web PKI ecosystem provides an underlying layer of security to many Internet protocols used today. By relying on Certificate Authorities (CAs), communication can be authenticated and encrypted based on a chain of trust. Unfortunately, this chain of trust has been broken in the past. For instance, in 2011, adversaries managed to issue fraudulent certificates on behalf of the DigiNotar CA, resulting in a loss of trust in DigiNotar. To better detect fraudulent certificates, Google introduced the concept of Certificate Transparency (CT), which is based on append-only logs that allow one to monitor and detect wrongly issued X.509 certificates.**

In this work, we investigate the potential of these logs as a data source for target reconnaissance. Concretely, we divide our study into two parts: First, we deploy several honeypot web servers over a period of 200 days to study the effect on incoming scanning traffic after pushing a certificate to one or more CT logs. We find that adding a certificate to a CT log leads to incoming network probes, just seconds after publishing the entry. This suggests that CT logs are used as input for web scans. In the IPv6 address space, our web server received 2,700 packets after pushing our certificate to a CT log, compared to 0 packets in our control group.

Second, we use large-scale active measurements to find potentially vulnerable domains from CT log data. Using certificate issuance and renewal patterns, we identify websites that are either at the beginning or at the end of their life cycle. Our results show that freshly deployed websites are not more likely to contain a known CVE compared to websites that just renewed their certificate. On the other side of the spectrum, however, we find that websites with an expired certificate, yet still deployed in the wild, tend to contain more outdated software, and hence more known CVEs. As such, CT logs can indeed function as a data source for target reconnaissance.

1. Introduction

It is commonly said that the Internet was never designed with security in mind. Rather, through the years, many enhancements and adjustments have been made to protocols and Internet policies to accommodate for various security shortcomings. While not perfect, these enhancements make it possible to keep the Internet functional and allow for its widespread use.

One of these enhancements was the introduction of a Public Key Infrastructure (PKI) on the Web, causing the advent of the Secure Sockets Layer (SSL) protocol, which is now better known as Transport Layer Security (TLS) [28]. TLS enables the encryption of network traffic (commonly done over TCP), as well as the authentication of the communicating parties. Not long after its introduction, many other protocols started supporting TLS to provide their services over an encrypted channel. For example, the Hypertext Transfer Protocol (HTTP), used for distributing web pages, developed its Secure variant (HTTPS), which makes use of TLS to authenticate a web server and encrypt its traffic to and from clients. To do so, each server requires an X.509 certificate, issued by a Certificate Authority (CA). These certificates provide proof of authenticity and include details for setting up an encrypted channel between client and server (see Section 2 for further details).

Unfortunately, the Web PKI contains some weak points. Because most CAs are run by people and operate on a foundation of trust, it is unavoidable that this trust can, and will, be broken. A notorious example is the infamous DigiNotar case of 2011. After an adversary gained access to DigiNotar's system, a wildcard certificate was issued for Google [12]. This enabled the execution of man-in-the-middle (MITM) attacks against Google services. Many more fraudulent certificates got issued due to this hack, and, shortly after, DigiNotar was no longer trusted by browser vendors and therefore ceased to exist [25].

To allow for faster detection of fraudulent certificates, Google started in 2013 an initiative called Certificate Transparency (CT), aimed at providing real-time monitoring of all certificates issued by participating CAs [17]. CT relies on numerous CT logs, operated by different parties such as browser vendors and technology companies, but also CAs themselves. The goal of these logs is to keep track of every issued X.509 certificate and make this list publicly accessible. While, in theory, everyone could push

a certificate to these logs, it is mostly CAs that do so. Specifically, when an organization requests a certificate, the issuing CA will push the certificate to one or more CT logs before providing it to the requesting party. CT logs are append-only and must meet strict availability requirements, as outlined in RFC 6962 [17]. By incentivizing CAs to push all newly issued certificates to multiple CT logs, companies can monitor these logs to detect any unwanted (and hence potentially fraudulent) certificates. The more CAs do this, the more difficult it becomes for an attacker to register a fraudulent certificate without being noticed by a CT log monitor (see Section 2 for more details on CT).

In its few years of operation, CT has proven to be successful at detecting wrongly issued certificates early on. For example, in 2016, Meta (then called Facebook) was able to identify, and later revoke, duplicate certificates for some of its own domains by actively monitoring CT logs [16]. The certificates were covering multiple subdomains of Meta, along with other domain names that were not in the company's control. The certificates were detected within an hour and immediately revoked.

While CT proves to be a step forward in battling the issuance of fraudulent certificates, there is unfortunately another side of the coin. By transparently pushing certificates to publicly accessible logs, sensitive information can be (unwillingly) leaked to the public. Previous work has shown how certificates from CT logs can leak employee details and confidential company data, and how this can be used for (spear)phishing attacks [32, 29]. Moreover, Kondracki et al. found several bot campaigns in the wild that actively scan websites and servers whose certificate was pushed to one or more CT logs [14].

In this paper, we describe how CT logs are a potential data source for target reconnaissance. Following the cyber kill chain, many adversaries precede their penetration by a reconnaissance phase, which is aimed at choosing potential targets for the attack [40]. While various methods exist for identifying potential targets (e.g., OSINT, port scanning), previous work did not take into account CT logs as an effective data source. CT provides the ability to continuously monitor its logs, making it trivial for an adversary to get a constant input of potentially interesting domains. Compared to global web scans, CT has a much lower processing time, especially for the IPv6 address space. In this work, we demonstrate how to extract potentially vulnerable websites from CT log data.

We conduct two types of large-scale measurements to scrutinize the potential of CT logs for finding vulnerable websites on the Web. First, we perform **passive** measurements on multiple honeypot domains to confirm whether a website receives more scanning traffic when its certificate has been pushed to a CT log. Using dedicated control groups, we capture the differences between websites whose certificate is included in one or more CT logs compared to websites with a self-signed certificate, and do this for both IPv4 and IPv6. We run our experiments for a period of 200 days, capturing longitudinal trends of these scanners. This allows us to get insights into their temporal patterns, which we demonstrate to be affected by certificate renewals. Additionally, we compare our results to honeypot data from a 2018 study by Scheitle et al. [32] to see how the behavior of CT log scanning has changed since 2018.

Second, we perform **active** measurements to detect domains that are more prone to be vulnerable. Concretely, we tackle this phase from two angles: The first aims at catching domains at the very beginning of their life cycle. We argue that, when deploying a website for the first time, off-the-shelf configurations or containers may be used, which might not have seen the latest security updates. Our second angle looks at the opposite side of the spectrum. Here, we argue that domains at the end of their life cycle may become forgotten, leaving them with outdated software [27].

By combining the results of our passive and active measurement studies, we are the first to empirically converge attack opportunities from CT logs with currently ongoing scanning activity, painting a comprehensive picture of the CT log threat landscape. In short, we make the following contributions:

- We deploy several web server honeypots over a period of 200 days and show that having a certificate in one or more CT logs leads to more incoming network scans compared to having no certificate in the CT ecosystem.
- We show that the effect of CT is especially noticeable in IPv6, where our control group received 0 incoming network packets compared to 2,700 in the CT-pushed group. By comparing our results to a study of Scheitle et al. in 2018 [32] we demonstrate that the general effect of CT has amplified over the last three years.
- Through large-scale active measurements, we are the first to use CT log data to detect potentially vulnerable websites for target reconnaissance. Our study shows that freshly deployed websites are not more likely to contain server distributions with a CVE compared to websites that have just renewed their certificate. However, we further demonstrate that CT logs can be used to detect actively deployed websites with an expired certificate. Additionally, we show that these websites tend to run more outdated software, and hence have more known CVEs. We conclude from this that CT logs can be used as a valid data source for target reconnaissance.

2. Background

This section explains how CT integrates in the Web PKI infrastructure, and how the process from requesting a certificate to embedding it into CT logs works.

2.1. Web PKI

X.509 certificates are the backbone of TLS/SSL, which is used by various protocols, such as HTTPS, to securely communicate over the web. Certificates are linked to public/private key pairs, enabling parties to establish an authenticated communication channel where traffic content is encrypted. This prevents the possibility of various network attacks, such as man-in-the-middle (MITM) attacks. Certificate Authorities (CAs) issue valid certificates after carefully validating the identity of the

requesting party. The CA then signs the certificate using its own self-signed root certificate. A root certificate can either sign another CA certificate, or a so-called "end-entity" certificate, which mostly belongs to a user or organization. CA certificates signed by a root certificate have the ability to also sign other certificates, creating a chain of signatures. This chain is more commonly known as the *chain of trust* since each signature provides a guarantee from the signing party that the signed certificate is legitimate and trustworthy. Since CAs can sign multiple certificates, verifying a certificate comes down to verifying a set of trustworthy CAs.

In HTTPS, Web browsers keep a list of trusted root certificates, called *trust store* or *root store*, which they use for verifying server certificates. Concretely, when a user visits a website, the client and the server bootstrap a secure connection by means of a handshake over TCP. In this handshake, the client verifies, among other things, the authenticity of the server by checking its X.509 certificate. This verification step is based on the chain of trust. Concretely, the browser will check whether the provided certificate is trusted (meaning, signed) by one or more root certificates in the browser's trust store. If so, the browser will trust the certificate and proceed to set up a secure connection. As a result, browsers can verify relatively quickly the trustworthiness of certificates, without needing extensive screening of the providing party.

Of course, if a trusted CA becomes compromised, the chain of trust is broken, enabling attackers to eavesdrop on encrypted communication. This fundamental flaw of web PKI is what inspired the advent of Certificate Transparency.

2.2. Certificate Transparency

Certificate Transparency (CT) enables public auditing of X.509 certificates, with the aim of detecting misbehavior and wrongly-issued certificates faster. This happens by using append-only logs (using Merkle trees) that collect certificates issued by CAs. Because CT logs are publicly accessible, anyone can track issuance activity from different CAs, and audit for inconsistencies or mistakes. Also, various monitoring services exist, which can be configured to keep track of specific domain names and check whether actors outside an organization try to register a malicious certificate for the organization's domain.

CT integrates into the web PKI ecosystem as follows: When an organization requests a certificate for a domain, the issuing CA pushes the certificate to one or more CT logs. Upon receiving a certificate, the CT log server creates a Signed Certificate Timestamp (SCT), which is a verifiable guarantee from the log that the certificate is, or will be, incorporated in the log. In addition, a Maximum Merge Delay (MMD) is provided by the log, indicating the maximum time it will take before the certificate is publicly visible in the log[1]. The CT log then sends the SCT back to the CA, which includes the SCT into the certificate of the domain before delivering it to the domain owner. The domain owner can then provide the SCT to requesting parties to prove that its certificate is included in a CT log.

[1]The MMD is usually 24 hours.

This last step has become more and more important, as many web browsers now require every certificate to be included in one or more CT logs. For example, Google requires in its Chrome browser that websites visited through HTTPS provide an SCT from a Google-operated log and from a non-Google-operated log [33]. Apple and Microsoft followed this approach shortly after by enforcing similar rules in their web browsers. This incentivizes domain owners to request certificates from CAs that push to CT logs, and subsequently also incentivizes CAs to push newly issued certificates to CT logs.

Although a second version of the CT protocol is described in RFC 6962-bis [18], no plans have been announced yet for its roll-out. As such, we build our work on the original protocol from RFC 6962 [17]. Nonetheless, our methodology and results should not be affected by the potential switch from version 1.0 to 2.0, as we rely on the general concept of CT for this work, and not its protocol-specific details. A change of protocol would therefore merely require an adaptation of our API calls to the log service.

3. Influence of CT Logs on Network Scanning

In this section, we explore the effects of CT logs on the incoming network traffic to a website. We first describe our methodology for generating domain names and including them in CT logs, as well as how we deploy honeypots behind these domain names to measure the differences in network traffic. Then, we analyze the results of our measurements and discuss how CT logs can influence scanning behavior. At the end of this section, we compare our results to a 2018 study by Scheitle et al. [32].

3.1. Methodology

Figure 1 depicts an overview of our empirical setup. First, we generate a random, non-guessable, fully qualified domain name (FQDN). We do so by using the `/dev/urandom` file of our UNIX system, which generates random content using the environmental noise of the hardware present in our machine [35]. We convert the content of this file into a string of 13 characters and prepend it to the domain name of our DNS zone (whose name we omit for double-blind review). The two strings, together, form a non-guessable FQDN. For example, if our randomly generated string is "`ugthyavcpleyt`" we construct the URL "`ugthyavcpleyt.domain.com`" and use it as a honeytoken for our experiments, where "`domain.com`" is the domain under our control.

Next, we request an X.509 certificate for our domain name using Let's Encrypt (LE) [1]. LE is a certificate authority that also runs its own CT logs [7]. As part of the process of issuing a certificate, LE makes sure the certificate gets pushed to the CT logs. As such, by requesting a certificate from LE we ensure that our newly generated domain name gets exposed to the public through CT logs. Furthermore, we make sure the certificate gets automatically renewed every 60 days. As a result, each time we renew our certificate, a new entry is pushed to the CT ecosystem.

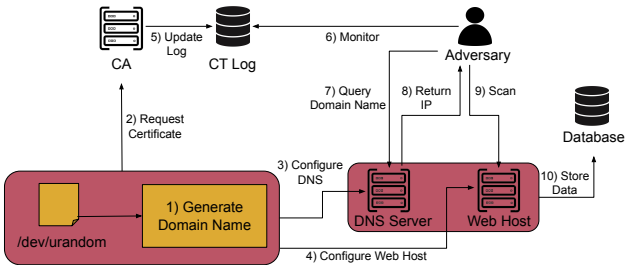


Figure 1: Diagram of our empirical setup. For simplicity, we depict only one instance, but, in reality, we perform multiple experiments with different configurations. Also note that we depict just one CT log, although in reality CAs can push to multiple logs.

While requesting an X.509 certificate for our domain name, we also add the necessary DNS entries to our DNS server (following RFC 1912 [3]) and set up a virtual host for our domain name. Both the DNS server as well as the virtual hosts act as our honeypot system. We capture all traffic received on both components and store the logs in our data server for analysis.

3.2. Threat Model

For this study, we assume an adversary with average infrastructure and resources, meaning an off-the-shelf PC and a regular Internet connection. Their motive is to find target websites to scan. While it would be feasible for the adversary to perform a scan of the entire IPv4 address space, such an approach requires more resources for repeated scans, and can easily produce overwhelming results that are time-consuming to analyze. Therefore, to perform a more efficient search, an adversary consults one or more CT logs to extract target domain names from certificates. All CT logs are publicly accessible by definition, so an adversary monitors them for newly added certificates. Each certificate contains one or more domain names, which an attacker then targets for scanning. Albeit monitoring services exist for CT logs, our methodology is not dependent on them, and hence this approach is feasible by accessing the CT logs directly.

3.3. Experiments

We design a set of experiments to scrutinize the effect of CT logs from different angles. When setting up the virtual host and configuring the DNS entries, we allocate some domain names to an IPv4 address, and others to an IPv6 address. We make sure that each IP address assigned to an experiment has never been used prior to this research.

For each experiment, we create an identical setup using a self-signed certificate rather than a certificate received from a CA. This prevents our certificate from being automatically pushed to one or more CT logs. We consider this group of experiments a controlled baseline for our evaluation, allowing us to compare measurements from a domain that has its certificate present in one or more CT logs, to measurements from a domain whose certificate is not present in any CT log.

Since we adhere to the standards of RFC 1912 to configure our DNS entries, each domain name is equipped with a PTR record on our DNS server. This PTR record exposes the domain name to actors scanning (random) IP addresses in DNS. These actors can then further scan the discovered domain names without consulting any CT log as part of their reconnaissance. Fiebig et al. have proven that such techniques are practical for enumerating hosts in the IPv6 address space using rDNS and PTR records [9, 10]. As such, we can expect to see incoming probes that were not influenced by an initial lookup in CT logs, but rather by the presence of a PTR record in our DNS zone. However, by launching a control experiment for each setup, we argue that this effect is accounted for. We see no reason to assume that the effects of these PTR records would differ between the control and the main experiment, allowing us to still compare the main experiment against the control experiment to isolate the effect of CT logs.

In total, we create four categories of experiments:

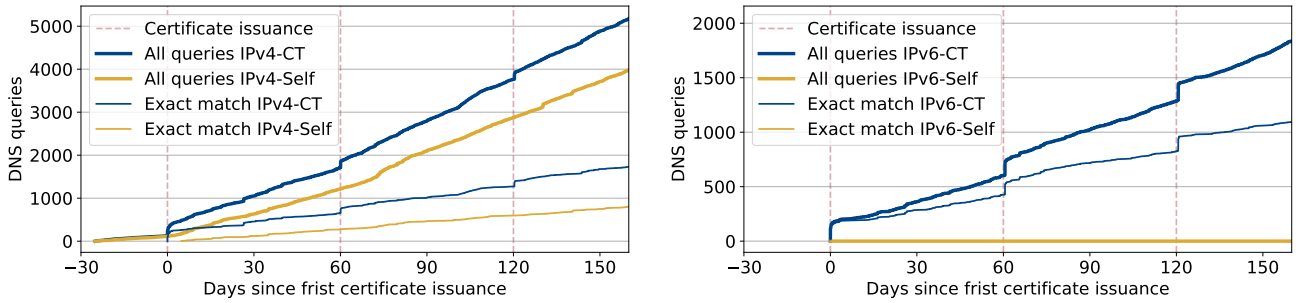
- **IPv4-CT:** In this experiment, we deploy a virtual host listening to one IP address in the IPv4 address space and host a domain that advertises an X.509 certificate requested from LE.
- **IPv4-Self:** This uses the same setup as IPv4-CT, but instead of requesting the certificate through LE, we self-sign the certificate to avoid it being pushed into the CT log ecosystem.
- **IPv6-CT:** In this experiment, we deploy a virtual host listening to one IP address in the IPv6 address space and host a domain that advertises an X.509 certificate requested from LE.
- **IPv6-Self:** This uses the same setup as IPv6-CT, but, instead of requesting the certificate through LE, we self-sign the certificate to avoid it being pushed into the CT log ecosystem.

We run the experiments between March 3rd, 2021, and September 5th, 2021 on Ubuntu 18.04.5 LTS using a one core CPU at 2.3 GHz and 2 GB of RAM. We use Nginx 1.14.0 and PowerDNS 4.1.1 to configure our web server setup.

3.4. Results

We start by analyzing the logs from our DNS server to see whether CT logs influence the number of requests received for the IP address of our URLs. Then, we look at how this effect is seen in the HTTPS traffic to our web server.

3.4.1. DNS Queries. Figure 2 depicts the cumulative DNS queries received over time for experiments IPv4-CT and IPv4-Self (Figure 2a), and experiments IPv6-CT and IPv6-Self (Figure 2b). During the setup of our experiments, we receive DNS queries stemming from the CA that are necessary for setting up the certificate. These queries were filtered out. The overall trend in both figures shows that our servers receive more DNS queries for domain names exposed through CT logs compared to the domains from our control group. We notice clear surges in the number of queries received each time a certificate for the domain gets pushed to the CT logs (i.e., every 60 days), either due to a renewal or a first-time issuance.



(a) IPv4 (b) IPv6

Figure 2: Cumulative number of DNS queries received by our servers over time, both for IPv4 and IPv6. We also depict the number of exact matches for our domain names received over time. In both address spaces, we clearly see how domains pushed to CT logs receive more queries, with significant surges shortly after pushing the certificate to the logs. It is also noteworthy that the IPv6 domains with no CT log entry do not receive any queries.

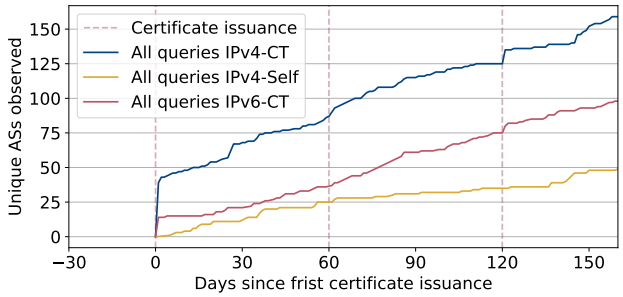
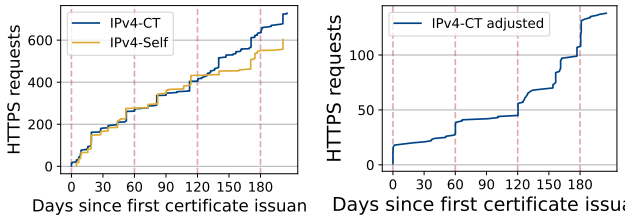


Figure 3: Number of unique ASs over time from which the observed DNS requests originate from. For the CT-influenced experiments, we observe a clear surge at the initial certificate registration, after which newly appearing ASs steadily increase over time. Note that we only depict exact matches here.



(a) Experiments (b) Adjusted

Figure 4: Cumulative number of HTTPS requests received by the domains over time. **(a)** depicts the total amount of incoming HTTPS traffic for both experiment IPv4-CT and experiment IPv4-Self. **(b)** depicts an adjusted version, where we subtract from IPv4-CT all sources stemming from packets in experiment IPv4-Self. In the latter, we again observe significant surges at the time of pushing our certificates to the CT logs.

Also when filtering for only exact query matches (that is, queries that fully match the generated domain name) we see an identical trend. This gives a first indication that domains with a certificate in one or more CT logs might receive more scanning traffic. Interestingly, we observe that our IPv6 control experiment receives no incoming DNS requests whatsoever. This can be attributed to the fact that, as of today, there exists no efficient method to fully scan the entire IPv6 address space in a feasible time frame. As we will discuss in 3.6, this demonstrates that CT logs show potential for scanning IPv6 hosts without needing to enumerate the entire IPv6 address space.

We receive requests from 2,291 unique IP addresses stemming from 267 Autonomous Systems (ASs) for our IPv4-CT experiment, and from 1,641 unique IP addresses in 163 ASs for experiment IPv4-Self. Similarly, we get requests from 959 unique IP addresses and 112 ASs for experiment IPv6-CT. For exact matches this becomes 1,016 IP addresses and 159 ASs for IPv4-CT, 455 IP addresses and 49 ASs for IPv4-Self, and 666 IP addresses and 105 ASs for IPv6-CT. The occurrence of a new IP address follows the same trend as the general pattern of requests depicted in figures 2a and 2b. That is, the number of unique IP addresses observed gradually increases over time, with significant surges at the certificate registration and renewal times. For the observed ASs, however, we notice a slightly different trend, which we depict in Figure 3 (note that we only show the exact matches). At the initial certificate registration, both the IPv4-CT and IPv6-CT experiment show a steep increase in newly appearing ASs, after which the trend resumes a stable pattern again. On the first certificate renewal, no clear surge is measured in newly appearing ASs. Interestingly, on the second renewal, we do in fact observe a clear increase due to the exposure of our certificate in the CT log ecosystem. This overall trend suggests that, although source IP addresses tend to differ per request, the incoming traffic seems to stem from a rather defined set of ASs. Nonetheless, the set of ASs is significantly larger for the CT-induced scans than for our control group.

3.4.2. HTTPS Requests. Figure 4a depicts the cumulative HTTPS requests received over time for experiments IPv4-CT and IPv4-Self. Although less apparent than in our DNS logs, we do see an eventual increase in requests to our experiment domain (IPv4-CT) compared to our control domain (IPv4-Self). Interestingly, when we filter out traffic in experiment IPv4-CT stemming from all source addresses belonging to the incoming packets in experiment IPv4-Self, the effect of CT becomes more apparent. We plot this effect in Figure 4b where we now

notice more clearly the influence of CT logs on incoming HTTPS traffic. Concretely, we again observe clear surges shortly after our certificate is pushed to the CT logs.

Similar to our DNS results, we detect no traffic on IPv6 for our control experiment IPv6-Self. As such, all incoming HTTPS requests in experiment IPv6-CT are likely due to the effect of pushing our domains to CT logs. In total, we receive around 2,700 packets destined for our server.

The incoming requests originate from 90 unique IP addresses stemming from 29 ASs, and 34 unique IP addresses stemming from 19 ASs for IPv4-CT and IPv4-Self, respectively. Similarly, for IPv6-CT, the requests come from 42 unique IP addresses and 9 ASs. Unlike the DNS data, we observe no clear patterns in source address statistics that could be attributed to CT log influence. This implies that renewing a certificate does not attract new sources of scanning.

3.5. Comparison to 2018 study

In 2018, Scheitle et al. studied the general deployment of CT and its influence on Internet traffic [32]. They suggest that, based on their observations, there might be actors in the wild who look for certificates in CT logs and scan the domains covered by the certificate. While their study focuses on the deployment and adoption rates in the CT ecosystem, not much attention has been paid to the difference between pushing a certificate to a CT log or not. Our experiments, therefore, improve upon the previous work in the following way: First, we run our experiments much longer, namely over a time span of 200 days compared to only 18 days in the previous study. We therefore capture longitudinal trends, giving us insight into the persistence of scanners, and allowing us to recognize interesting temporal patterns. As such, we are the first to report on the effects of certificate renewals, which show up in our data as a relevant factor for scanning behavior. Specifically, we observe surges at each renewal, showing that websites are continuously affected by scans due to CT, and not just by one-time probes.

Moreover, based on prior work, it was unclear whether the additional scanning traffic caused by CT would, over time, grow in volume, potentially to a problematic size. Our measurements show that these effects are fairly mild, concluding that additional traffic should not be a concern for website administrators.

Second, Scheitle et al. mention that adequately discerning arbitrary scanning from informed scanning was not possible in their experiments, making it difficult to purely attribute their results to the effect of CT. To overcome this challenge, we introduce control groups in each of our experiments. Since our control groups are identical (with the exception of a self-signed certificate instead of a CA-issued certificate) in setup compared to the regular experiments, we can adequately observe the direct effect of CT logs. In particular, for the IPv4-CT experiment, we use this data to reveal the surges caused by CT-induced scanners (see Figure 4b).

Third, Scheitle et al. create no DNS PTR record for their domain names, therefore not strictly adhering to the standard described in RFC 1912. The authors argue that this prevents the influence of rDNS walking on the results of the experiments. We do include a PTR record in our DNS configuration for each domain, and combat the influence of rDNS walking by again using control experiments for each setup (see Section 3.3 for more detail).

TABLE 1: Comparison of our results to a 2018 study by Scheitle et al. [32]. Over a period of 3 years, we notice a drastic change in the influence of CT logs on scanning behavior. The detection of the first DNS query in IPv4 has gone down by more than 50%, and the detection of the first HTTPS request has a speed up of 97%-99%. Also in IPv6, we notice a significant change, with the previous study detecting 0 incoming packets whereas we detect 2,700 incoming packets.

	2018 [32]	**2021**
First DNS query (IPv4)	73s-197s	23s
First HTTPS request (IPv4)	3,540s-1,641,600s	84s
Number of IPv6 packets	0	2,700

Despite these differences, we still see value in comparing the results of Scheitle et al.'s early exploration on this topic to capture the change of landscape since 2018. Table 1 highlights some striking changes between the results of our study and the study of Scheitle et al. (Note that we use the same CA, pushing to the same CT logs, making the comparison sound.) Overall, we conclude that the influence of CT logs on scanning traffic has significantly increased since 2018. For example, the first incoming requests arrive much faster compared to the 2018 study. For DNS in IPv4, this is a speedup between 68.5% and 88.3% for the first query received. For HTTPS in IPv4, this difference is even more pronounced, with a 97.6% to 99.9% faster detection time in our study compared to the 2018 study. As for IPv6, the previous study by Scheitle et al. detects zero incoming scanning traffic for their domain. Over the time span of three years, this has significantly increased, with our study detecting 2,700 incoming packets for IPv6. Finally, by including dedicated control groups in our experiments, we strengthen the claim that these effects are a result of pushing a certificate into the CT ecosystem, as opposed to just arbitrary scanning.

3.6. Takeaways

In this section, we explored the influence of CT logs on incoming scanning traffic. We conclude that pushing a domain's X.509 certificate to one or more CT logs indeed increases the incoming network traffic. Specifically, we observe that more DNS queries are made, with significant surges around renewal days, both for IPv4 and IPv6. Also for HTTPS, we notice the same behavior. Most notable is the influence of CT logs on incoming IPv6 traffic. Our experiments show that having a domain's X.509 certificate in one or more CT logs can increase the incoming IPv6 traffic from 0 to several thousand of packets. Given the extreme difficulty of scanning the entire IPv6 address space, these results indicate that CT logs are a viable input source for IPv6 network scanning. By comparing our results to a study from 2018, we also show that the effect of CT logs on scanning traffic has amplified over the last three years. As such, we conclude that CT logs

are indeed used in the wild to scan domains, and will only become more popular in the future.

Next, we explore whether using CT logs as a reconnaissance technique is actually worthwhile.

4. Detecting Newly Deployed Domains in (Near) Real Time

Although the standard MMD for a CT log is 24 hours, we discovered from our honeypot experiment that a certificate often gets pushed much sooner into the CT ecosystem. This led to very early scans of our domain, even though our website was only deployed for merely seconds. In this section, we take the perspective of an adversary and use several CT logs to perform a reconnaissance scan. In particular, we are interested in websites that were just deployed, as we hypothesize that many developers rely on off-the-shelf containers or legacy setup files to launch their new websites. While practical, such configurations might install outdated software, increasing the probability that a web application lacks the latest security defenses. By leveraging the short time window in which a certificate is pushed to a CT log, we attempt to identify vulnerable domains before they can update any outdated server software.

4.1. Methodology

To determine whether a domain is freshly deployed, we use their X.509 certificate as a proxy. If a domain appears for the first time in a certificate, we assume it is being deployed for the first time as well. Using Certstream [4], we actively monitor newly pushed certificates in CT logs. Certstream regularly downloads the Signed Tree Head (STH) of CT logs and checks when the size of the log changes over time. If the tool detects a change in size, it retrieves the latest added certificates, parses them, and outputs the domain names that are covered by the certificate. We run Certstream for around 24 hours on July 19^{th}, 2022. We probe the URL of each domain name we encounter. Note that we do not include a specific `index.html` or similar suffix. We merely take the name that was included in the certificate. In case of a wildcard, we skip the entry. For each probe, we call the HTTP `HEAD` method. In case of a redirect, we store both the initial HTTP Header and the final redirected-to Header. In case of an error or timeout of longer than 30 seconds, we reprobe the URL. After 3 probes, we consider the website unresponsive. In total, we collect probes for 662,562 URLs.

Because a domain name can be added to multiple certificates, we remove all duplicate entries from our results, leaving a total of 501,791 entries. Then, we probe all responsive websites 5 times over the course of a month: 1 day after, 3 days after, 1 week after, 2 weeks after, and 1 month after the initial probe. We use the same probing mechanism in case of a timeout.

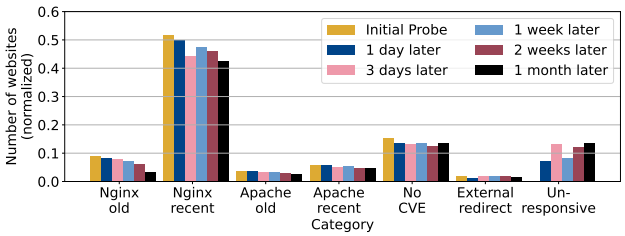
Next, we determine which domains appeared for the first time in a certificate, and which appeared multiple times as a result of a certificate renewal. For this step, we rely on the Censys certificate database [6]. For each domain, we check how many certificates it has in the Censys database. If it has more than one, we assume the domain has not been deployed for the first time. Unfortunately, due to API limits, we are not able to check every URL in our set. As such, we prune our set of URLs by focussing only on the websites using one of the two most used server distributions, namely Nginx and Apache [30]. Furthermore, we only keep the Nginx and Apache instances that expose their version number in the HTTP Header since we will use this as a basis for our evaluation (see Section 4.2). This trims our set down to 25,319 URLs. Then, we use the CT logs of LE to pre-remove certificates. LE runs its own CT log infrastructure called Oak, which they use to push newly issued certificates [7]. Because LE is one of the most used CAs [36], we assume many of the domains in our set have a certificate pushed to a log in Oak. We download all Oak logs and check whether a domain has more than one certificate in the logs. If so, we can be certain it has also more than one certificate in the Censys database, meaning we do not have to check it through the Censys API. If a domain has only one certificate or less in the Oak log we do check it with Censys. Note that it would be possible to download more CT logs to make the pre-removal more robust. However, this is paired with a significant increase in storage requirements. For our data, the Oak logs were sufficient to prune our set of URLs to an acceptable size. After checking our pruned set of URLs in Censys, we find 986 domains that are deployed for the first time and 24,333 that appeared due to a renewal of their certificate.

4.2. Security Evaluation

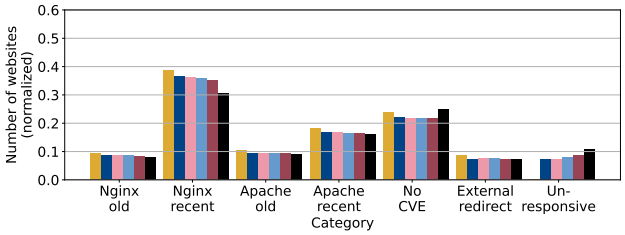
We evaluate all encountered server versions against 2 CVEs per server software. For Apache, we pick **CVE-2018-17189** [21] and **CVE-2021-44790** [24] and for Nginx we pick **CVE-2018-16844** [20] and **CVE-2021-23017** [23] . Notice that for both server software we have a recent CVE (2021) and a less recent CVE (2018) to account for the general delay between the release of a CVE and an administrator updating their system accordingly. Furthermore, all CVEs are retroactive, meaning they apply to all versions below a certain number.

Figure 5a shows the results of our analysis on websites deployed for the first time. What stands out immediately is that some websites become unresponsive after the initial probe, even after reprobing the URL 3 times. Due to this unresponsiveness, many websites that run a vulnerable version of their server software are merely online for a brief moment at the start of their deployment. This short time window could be attractive to adversaries for executing attacks on websites that might not be fully ready for production yet.

Interestingly, we also observe an increase in unresponsiveness for the group of domains that simply had a renewal of their certificate. Figure 5b shows this trend. Also here, the number of vulnerable versions seems to decline over time as a result of this. However, the trend is different over time. Where the renewal group has a steadily increasing number of unresponsive websites, the group of websites that are deployed for the first time has a less structured pattern. The pattern suggests a "testing-out phase," where administrators put the website in production to test some functionality, and then take it offline again



(a) Websites deployed for the first time



(b) Websites who renewed their certificate

Figure 5: Categorization based on server software. Figure **(a)** depicts the results for the group of websites that appeared for the first time in a certificate. Figure **(b)** depicts the results for the group of websites that renewed their certificate. "old" and "recent" signify the 2018 and 2021 CVE for each software, respectively.

to continue development, potentially repeating the process several times.

Contrary to the group of first-time deployment, in the renewal group, we notice that the set of versions *without* a CVE *does* increase, meaning that the change in landscape is not purely due to websites becoming unresponsive. In the initial probe, 23.8% did not have a version with a CVE, which after one month increased to 24.9%, showing that websites have patched their outdated software. For the first-time group, this number actually decreases from 15.3% to 13.5% due to the unresponsiveness of the website. This suggests that for the websites that are deployed for the first time, administrators do not update their software right away, but rather take the website offline, presumably to continue further development. Furthermore, the first-time group seems to contain more recent CVEs whereas the renewal group contains more of a mix (although still favoring more recent CVEs). This makes sense, as we would expect the renewal group to indeed contain websites that have been deployed for a longer time, and therefore might not have seen as many (security) updates. This "legacy effect" might suggest that it is more beneficial to look at CT logs for the detection of old, unmaintained (and therefore potentially less secure) websites, rather than newly deployed ones. In the next section, we explore this idea in further detail.

5. Detecting Domains with Expired Certificates

Because CT logs are append-only, we can get an insight into the certificate renewal patterns of websites. That is, we can detect if a domain owner stopped renewing their certificate if the last known certificate of a domain has expired, yet the domain name no longer appears in newly added certificates from the CT log. By probing this set of domains, we can verify whether they are indeed advertising an expired certificate, or if they simply stopped pushing to the CT log (due to, for example, a change of CA).

5.1. Google Pilot Log

As a starting point, we take the Pilot log from Google. Pilot is one of the initial CT logs, and remained active even at the time of our experiments[2]. Earlier work has shown how Pilot was by far the most pushed-to log in the early years of CT [11]. Since we look for domains with an expired certificate, we want to maximize our data set for older domains, i.e., domains added in the initial years of CT. As such, we deem the Google Pilot log as the best fit for this study.

We download the complete log on October 19th, 2021 when the `tree_size` of Pilot is 1,077,308,203. After parsing the raw data for each entry, we obtain the full X.509 certificates that were pushed to the log. Many companies, like Google and LE, use test domains to validate that their logs are operating correctly. As such, many of these domains, like "flowers-to-the-world.com" from Google and "woodpecker.testing.letsencrypt.org" from LE, appear frequently in our data set. We consider these domains irrelevant to our study and hence remove all their certificates from our data. For the remaining certificates, we make a separate entry with the start and end date of the certificate for every domain it covers. This gives us 4,705,267,788 entries in total.

5.2. Extracting Domains with a Potentially Expired Certificate

From our list of domains, we are interested in the set of unique domain names whose last-seen certificate has passed its expiration date. Since each time a domain name renews its certificate the CA pushes it to the log, our list contains duplicate names, each with a different expiration date. We filter out these duplicates by keeping the entry with the latest expiration date. Then, we take as a threshold September 1, 2021, 12:00:00 PM and split the list in two: the set of entries with an expiration date before our threshold (called Exp) and the set of entries with an expiration date after our threshold (called $Valid$). Even though this step was performed in December, we pick September as a threshold to leave some margin for renewals that are only a few weeks late. Finally, we obtain our desired set of domains (called $Probeset$) by picking all domains that are in the Exp set, but not in the $Valid$ set, i.e.:

$$Probeset = \{d \mid d \in Exp \wedge d \notin Valid\}.$$

Since many certificates contain a wildcard entry, we use passive DNS data to resolve the wildcards into an accessible URL.

[2]Although Pilot was active during the time of our experiments, Google has announced since then the planned retirement of the log for May 1st, 2022 [26].

5.3. TLS Probing

We use ZGrab 2.0 [41] to capture the current certificate present on each domain if any. ZGrab can be configured to perform a TLS handshake and store transcripts of the exchange including, among other details, the X.509 certificate and its expiration date. Using this, we can check if the domains are indeed advertising expired certificates, or if they renewed their certificate without pushing to the same CT log as before.

We launch our scan on December 16^{th}, 2021. Unfortunately, due to network issues, our scans got prematurely ceased on December 21^{st} after completing 77% of our probes, which equates to 1,192,725,834 domains. We report that 762,462,458 (64%) advertise a non-expired certificate, while 11,032,644 (1%) contain an expired certificate. 418,229,732 (35%) of the domains either timed out or did not exist anymore.

5.4. Security Evaluation

We evaluate the security posture of the domains in our set in three different ways. First, we call the HTTP `HEAD` method for each domain, extract the server software HTTP Header, and compare it against known CVEs, as we did in Section 4.2. Second, we perform a similar analysis but for the website's Content Management System (CMS) by scrutinizing the source code of each index page. Lastly, we look for other indicators in the source code that signify bad security posture based on previous work.

To baseline our results, we create an additional set of domains to use as a control group during the security evaluation. Concretely, we randomly take 11,032,644 domains from our initial ZGrab scan that responded with a non-expired X.509 certificate. As such, we obtain an equal size group of domains that can be compared to our set with expired certificates.

5.4.1. HTTPS Headers. For both groups, we call the HTTP `HEAD` for each entry, and extract the server software if included. We use `curl` to grab the `Server` field in the HTTP Header. Furthermore, we store the HTTP Status Code to filter bad responses. We perform our scan on January 13^{th}, 2022 for the expired group, and on January 15^{th}, 2022 for the control group.

Figure 6 shows the distribution of HTTP response codes for both groups. Indeed, our expired group produces significantly more failed responses compared to the control group. Presumably, these domains have been taken offline, explaining why they are no longer present in the CT log.

Additional to the status code, we extract the `Server` field from the header. Such field is present in 7,389,804 (85%) responses from the expired group, and in 9,445,172 (90%) from the control, not considering the failed responses. We again notice in our data that Nginx and Apache are the most popular used server software. As such, we focus again on these two server applications for our analysis, using the same CVEs as in Section 4.2.

Although many servers advertise the software used to host their domain, not all of them advertise the actual version number. Here we notice an interesting discrepancy between the expired group and our control group. In the expired group, 717,327 (28%) of the Apache servers give away their software version, while for the control group this is only 184,423 (14%). Similarly, for Nginx, we observe that 1,342,084 (51%) show a version number in the expired group and only 379,826 (8%) in the control. While not a direct security measure, hiding the software's version number does slow down an attacker, as it requires them to perform extra, more advanced analysis of the website to assess whether it is running vulnerable software. In this regard, it is interesting to see that the control group shows better practices.

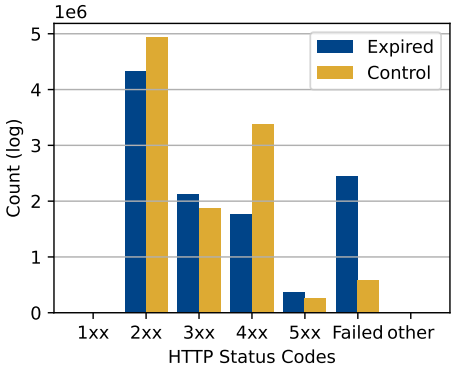


Figure 6: HTTP Response Codes for both the expired and control group. As expected, the expired group produces many more failed responses, presumably due to the website being no longer reachable.

Table 2 summarizes the results[3]. For Apache, we find that the expired group is more susceptible to both CVEs compared to our control. Moreover, the discrepancy between the two groups becomes bigger when we focus in particular on the CVE from 2018, with the expired group showing that 69.19% of the hosts are vulnerable and in the control group only 48.95% of the hosts are vulnerable. Also, the total percentage of vulnerable instances in both groups is significantly higher for the more recent CVE. This indeed shows that the more recent a CVE is, the more likely it is to find it in the wild since administrators had less time to patch the software. For Nginx, we see an almost identical pattern, with the 2018 CVE being present in 56.20% of the hosts in the expired group, compared to 34.39% of the hosts in our control group.

As such, we conclude that for both server distributions, domains with an expired certificate have a higher probability of running a software version that is susceptible to a known CVE. Consequently, CT logs have the ability to expose these domains to the public, making it easier for adversaries to gather a potential target set during their reconnaissance phase.

5.4.2. Content Management System. Instead of building a web page from scratch, many developers make use of a Content Management System (CMS) to generate the source code of their pages. According to the HTML

[3]Note that the distributions differ from the distributions in Figure 5. This is due to version updates that happened in between these two experiments (e.g., Nginx went from 1.21.5 to 1.23.1). We assumed this might incentivize administrators to update (and our data shows they do), and hence did the extra scan to construct a fair control group for each experiment.

TABLE 2: Results of comparing the software version of Apache and Nginx instances against known CVEs. We observe that in the expired group more instances are vulnerable to known CVEs compared to our control.

	Apache		Nginx	
	CVE-2018	**CVE-2021**	**CVE-2018**	**CVE-2021**
Expired	69.19%	79.31%	56.16%	89.47%
Control	48.95%	73.42%	34.36%	87.47%

standard [37], the software used to generate the web page should be listed in the `meta` tag, along with an accompanying `content` field as follows: `<meta name=''generator'' content=''MySoftware 1.2.3''/>`. As such, we use regular expressions to look for this pattern in the source code of the index page and extract a CMS (with its version number), if present.

Before scrutinizing the source code, we prune our set of domains to preemptively discard websites we are not interested in. First, we remove all websites that did not respond with an HTTP status code of `200 (OK)`. Second, we discard the domains from our previous analysis that were already identified as having a vulnerable software version. In total, this brings our expired set down to 3,609,833 domains and our control group to 4,816,749 domains.

We download the source code of the index pages on January 21^{st}, 2022 for the expired group, and January 23^{rd}, 2022 for the control group. Of course, some domains might still display a "404 Page Not Found" or similar, even though they respond with a status code of `200 (OK)`. To account for this, we search for keywords in the source code and filter out `ERROR` pages accordingly. This leaves our expired set at a size of 2,355,193 domains, and our control at 1,519,777 domains.

We successfully extract a CMS from 223,810 (9.5%) domains in our expired group, and from 192,846 (12.7%) domains in our control group. Not unexpectedly, the most popular software in our set is WordPress, both in the expired group (98,890 instances) as well as in the control group (76,149 instances). Consequently, we compare the version number of detected WordPress instances against a known CVE. Specifically, we look at **CVE-2019-8942** [22], which is again not overly recent to account for updating delays. Interestingly, just as with the server software, we find that the control group tends to hide more actively the actual version number as opposed to the expired group: We successfully infer a version number in 94,360 (96.4%) of all WordPress cases in the expired group, while only 49,118 (64.5%) in the control group.

Table 3 summarizes the results. We again observe that the expired group contains more vulnerable version numbers compared to our control group, with 19,618 (20.79%) cases for the expired group, and 5,864 (11.94%) for the control group. However, much of WordPress' popularity comes from the various plugins designed to extend the CMS. Some of these plugins, including their version numbers, are advertised in the downloaded source code samples. As such, we also compare known vulnerabilities for two specific plugins [38, 39] to their version numbers. We again notice the same trend, with the expired group containing more vulnerable versions for "Download Manager" (109/128 = 85.16% expired, 26/73 = 35.62% control) as well as for "Layerslider" (22/184 = 11.96% expired, 16/194 = 8.25% control). The combined total of these numbers does not, however, influence the overall percentage of WordPress due to the small presence of these plugins in our data set.

TABLE 3: Results of comparing WordPress software versions, as well as those of two WordPress plugins: Download Manager and Layerslider. We observe that in the expired group more instances are vulnerable to a known CVE compared to our control.

	WordPress	**Download Manager Plugin**	**Layerslider Plugin**	**Total**
Expired	20.79%	85.16%	11.96%	20.86%
Control	11.94%	35.62%	8.25%	11.96%

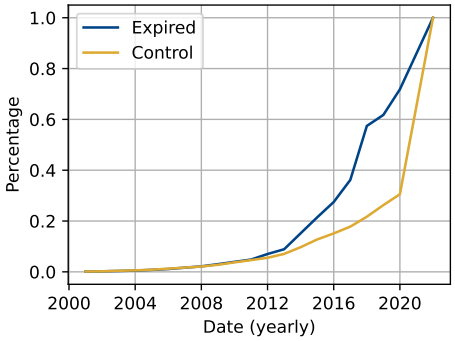


Figure 7: CDF of the extracted copyright years in both groups. The expired group has significantly older copyrights, which might be an indication that these are not actively maintained anymore.

Overall, we reached the same conclusion as with assessing the version numbers of server software: The group with expired certificates has significantly more vulnerable version numbers compared to our control group. We, therefore, show again that CT logs can indeed expose these domains to the public, requiring only a little effort to construct a set of potentially vulnerable websites.

5.4.3. Outdated Copyright Statements. Prior work has shown that outdated copyright statments on a web page can indicate that the page is no longer maintained [27]. Moreover, these pages tend to be more vulnerable to trivial exploits, such as Cross-Site Scripting and SQL Injection. As such, we extract the copyright statements from the source codes of our web pages. In the expired group, 442,143 websites (18.8%) displayed a copyright statement, and in the control group, 196,456 websites (12.9%) did. Figure 7 shows the Cumulative Distribution Function (CDF) for the encountered copyright years in both the expired group, as well as in the control.

Our copyright analysis shows that in the expired group, 70% of the index pages have a copyright statement year of 2020 or older, showing that the majority seems to be outdated. For the control group, on the other hand, we notice that only 30% have a copyright statement year of 2020 or older, meaning they indeed appear to be more actively maintained.

6. Discussion

In what follows, we discuss the implications of our results and question whether this should affect the deployment of CT. Furthermore, we highlight the ethical aspects of our experiments and how we address them. Finally, we explain the limitations of our work, and suggest future improvements and directions to build upon our research.

6.1. Advantages of Certificate Transparency for Target Reconnaissance

In Section 5 we showed how CT logs can be leveraged to find potentially vulnerable targets in the wild. Attackers might be incentivized to use this approach since it requires less time and resources compared to, for example, global web scans. This is especially true for IPv6, where no methodology exists to efficiently scan the entire address space. Using current techniques, scanning the entire IPv6 address space for potential targets would take roughly $2 \cdot 10^{25}$ **years** [31]. In contrast, with CT, an adversary eliminates the need for a global port scan since they can directly tap into a source of potential targets from the logs, which we demonstrate with our active scans. When looking for expired domains (Section 5) the biggest bottleneck is downloading and processing a CT log. For the Google Pilot log, this took us less than 24 hours. When looking for newly deployed domains (Section 4) the speedup is even more dire since public streams (such as Certstream [4]) provide a constant flow of newly appearing certificates. It is therefore no surprise that our passive measurements (Section 3) observe early probes nearly seconds after obtaining a X.509 certificate, demonstrating that it is not only possible to circumvent the limitations of global web scans, but that this is in fact being used in the wild. With our study, we are the first to empirically correlate these attack opportunities from CT logs with currently ongoing scanning activity, painting a comprehensive picture of the CT log threat landscape

6.2. Abolish Certificate Transparency?

Introducing ways of abusing CT for malicious purposes might suggest the desire for its removal, both as policy as well as a technology. While it is true that CT introduces a novel data source for target reconnaissance, we believe the benefits of CT still outweigh the small costs. For example, we demonstrate in the first part of this paper how pushing a certificate to a CT log increases your incoming network traffic. Specifically, we identify surges almost immediately after the issuance and renewal of a certificate. However, while significant enough to distinguish them from non-CT-related traffic, these surges were never of a volume that would take down a server or a network. As such, even small-scale commodity servers should be able to handle the additional traffic that CT causes.

In the interest of completeness, we do mention that, from a theoretical point of view, a server could experience heavy load due to CT-related traffic if it hosts multiple domains, all of which issue/renew their certificate at the same moment. In such a scenario, the surges might coincide and accumulate to a potentially problematic size. Nonetheless, we estimate the probability of this happening to be low.

In the second part of this paper, we describe two approaches to how CT logs can potentially be used for target reconnaissance. Again, we see no significant added harm caused by CT directly. Moreover, the issue of a website running outdated software is not an effect of the CT ecosystem. Rather, it is the responsibility of the administrator to make sure their software is up-to-date according to the latest security standards.

However, although we argue that the costs of CT do not outweigh its benefits, we simultaneously show that scanning effects have increased since 2018. As such, despite the fact that the current state of the CT ecosystem does not introduce any significant harm to the aspects we study, we do think further monitoring should be done to prevent small issues from escalating to a more problematic state.

6.3. Ethics

Internet measurement studies and security research in general require careful evaluation of their ethical implications. For this work, we base ourselves on the Menlo report [2]. In the honeypot experiment, we see no harm in our design choices. We perform passive measurements that simply host a domain and log the incoming traffic, which is no different behavior from any regular website hosted on the Web today. As for the active measurements, we do rely on best practices. Concretely, we provide every probe with a customized User-Agent containing contact details to opt out of our study. Additionally, we set up a TXT record with contact information, available through a reverse lookup of any of the IP addresses from which we scan. During our experiments, we received one request to opt out of our study, after which we removed the domain from our data and future scans.

Concerning the security evaluations of this study, we stick to the reconnaissance phase of the cyber kill chain [40]. This entails that we merely scan for software versions –and compare them against a CVE database– without advancing further to the actual exploitation of any of the found CVEs. As such, we are able to show the potential impact of CT without causing harm to a server or website in the process. We do note, however, that an adversary could take further steps after identifying targets but considered the execution of this harmful and unnecessary for the results of our research.

6.4. Limitations and Future Work

For the honeypot, we use certificates containing concrete domain names rather than wildcard entries. Many organizations include multiple domain names in a certificate, simplifying the process through wildcards (e.g., `*.example.com`). This does not allow an adversary to immediately identify all websites protected by the certificate, therefore preventing them from using the websites as targets. Future work could investigate whether this affects scanning behavior and whether adversaries then rely on additional sources, such as for example passive DNS data, to match the wildcard entries with concrete URLs.

To detect domains with an expired certificate, we solely rely on the Google Pilot log. Although this was the most active log in the early days of CT (therefore providing more old certificates compared to other logs from that period), many more CT logs exist. To keep processing time reasonable, and demonstrate how with moderate storage (order of 100GB) one can perform our methodology, we focus on only one CT log. By scrutinizing more logs, the set of potential targets could be expanded. However, this expansion would require more storage and processing resources. Our results can therefore be seen as a lower bound on the number of target websites that can be found via CT log data.

7. Related Work

Although Certificate Transparency is a fairly recent concept, the scientific community has already looked at the topic from different angles, such as deployment, security, and privacy. Gustafsson et al. were the firsts to give an overview of CT deployment through active and passive Internet measurements [11]. They collect data from several CT logs, and compare the entries to encountered certificates in the wild. Albeit the study was performed before major browser vendors started enforcing CT policies, the authors conclude that CT logs contain a representative set of certificates compared to their passive monitoring data. Two years later (after CT policy enforcement by web browsers), Google published a study with similar conclusions, showing that already 63.2% of HTTPS connections in Chrome contain certificates present in two or more CT logs [34]. Furthermore, the authors study the effect of non-compliance with CT policies on end-user behavior, and show that users do react unsafely when encountering breakage due to failure of CT compliance by a website, therefore incentivizing websites and CAs to adhere to CT policies and standards.

Also, the work of Korzhitskii et al. studies the deployment of CT logs, focusing on how the root store ecosystem is shaped among different logs and in comparison to browser trust stores [15]. Like browsers, CT logs keep a list of root certificates that are trusted to sign other certificates. Logs can then enforce each pushed certificate to contain one of their root certificates in the chain of trust. The paper highlights many discrepancies between browser root stores and CT root stores, along with concerns such as non-sufficient coverage of roots by CT logs compared to software, and the inclusion of compromised root certificates in CT logs.

Besides deployment, large-scale measurements have also been used to study the security of CT. Scheitle et al. were the first to do so, and provide a preliminary evaluation of some security implications caused by CT [32]. Similar to our study, the authors deploy a honeypot to examine the effects of CT presence on website scanning behavior. As mentioned earlier, we improve upon their design and show how the effects of CT logs have amplified over the years (see Section 3.5 for more detail). The paper also demonstrates how CT logs leak DNS information and describe a methodology for subdomain enumeration using CT data. Roberts et al. further study information leakage in CT logs [29]. By searching (sub)domain names in certificates, the authors are able to leak sensitive information such as employee names, user names, email addresses, and confidential company data such as unreleased products and campaigns. The paper argues that such information can easily be used for targeted attacks such as spearfishing or social engineering. Finally, Kondracki et al. deployed a distributed honeypot system to identify and classify CT bots [14]. They found various campaigns actively using CT logs to scan servers in the wild. Several of these campaigns showed clear malicious intent, for example by automatically launching Log4J exploitation attacks.

CT log data can also be used to detect phishing websites. Because traditional blocklists experience a delay between the deployment of a phishing domain and its inclusion in a blocklist, CT offers a closer to real-time detection opportunity by advertising newly deployed domains much quicker. Fasllija et al. present *Phish-Hook*, a tool to detect phishing websites in real-time using CT logs [8]. The authors leverage machine learning to classify each website in one of five risk categories rather than producing a binary conclusion (i.e., phishing vs. non-phishing). *Phish-Hook* relies solely on CT log data and therefore does not require further source code analysis or access to network traffic data. With this method, over 90% of phishing websites could be correctly identified. However, the authors only apply their method on a pre-existing sample set of URLs and merely hypothesize that the method can hence be applied to CT logs. Unfortunately, such an evaluation is not provided in the paper. Drichel et al. go beyond a theoretical solution and developed a full pipeline for detecting phishing websites using CT logs [5]. Primarily, the pipeline is designed for machine learning approaches to detect URLs in certificates that belong to phishing websites. The pipeline is modular and open-source, allowing for further expansion of their techniques. Furthermore, the authors use their pipeline to develop ground truth data for past CT log entries that can be used to train future phishing detection solutions. Also, Marquardt and Schmidt make use of CT log data to extract data sets for research related to domain names [19]. Instead of focusing on machine learning, the authors developed a pipeline to extract input domains for large-scale measurement studies, aiming to complement other input lists such as the Alexa top 1M and the Majestic top 1M. Their analysis shows that such a CT log data set contains similarities to other top lists (such as outdated software distribution), but also some differences (such as a higher DNS error rate for the CT-based list).

8. Conclusion

Certificate Transparency is a relatively new concept that aims to solve fundamental trust issues in the Web PKI ecosystem. It has been extremely successful so far in seamlessly integrating into the Web PKI without breaking the web or causing major frustrations. While it has been studied from many perspectives, such as deployability, privacy, and potential for phishing detection, not much work has been done on the potential of CT logs as a data source for attacks.

In this paper, we used both active and passive measurements to shine a light on this blind spot. Through our honeypot system, we investigated the difference in network traffic between websites with and without a certificate

in one or more CT logs. After running our experiment for a period of 200 days we conclude that CT does indeed increase the number of incoming probes to a server, suggesting that actors use these logs as input for web scans. Compared to a previous study from 2018 we notice that the first scans arrive much quicker (merely seconds after certificate issuance) and that it causes significantly more traffic for IPv6 hosts compared to websites with a certificate that is not pushed to a CT log. Using active measurements, we captured websites both at the beginning as well as the end of their life cycle. We show that CT log data does not provide an advantage for detecting more vulnerable domains at the beginning of their life cycle, as it gives no significant difference over domains that just renewed their certificate. As a result, we report on a new approach for finding potential target websites. Specifically, we looked at websites that stopped renewing their X.509 certificate, yet continued running (part of) their infrastructure in production. We show that this group has significantly more server software with a known CVE compared to our control group. As such, we conclude that CT logs can function as a potential data source for target reconnaissance.

9. Acknowledgement

We would like to thank the anonymous reviewers for their insightful feedback and comments. Furthermore, the authors would like to thank Frans Broos for arranging needed infrastructure and general support of the project.

This material is based upon work partially supported by a gift from Google on Security, Privacy and Anti-Abuse. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of Google, or the author's host institutions.

References

[1] Josh Aas et al. "Let's Encrypt: An Automated Certificate Authority to Encrypt the Entire Web". In: *Proceedings of the 26th ACM SIGSAC Conference on Computer and Communications Security (CCS)*. Ed. by XiaoFeng Wang and Jonathan Katz. London, United Kingdom: ACM, Nov. 2019. ISBN: 978-1-4503-6747-9. DOI: 10.1145/3319535.3363192.

[2] Michael Bailey et al. "The Menlo Report". In: *IEEE Security & Privacy* 10.2 (Mar. 2012), pp. 71–75. DOI: 10.1109/MSP.2012.52.

[3] D. Barr. *Common DNS Operational and Configuration Errors*. Feb. 1996. DOI: 10.17487/RFC1912. URL: https://www.ietf.org/rfc/rfc1912.txt (visited on 04/16/2022).

[4] CaliDog. *Certstream Server*. Aug. 12, 2022. URL: https://github.com/CaliDog/certstream-server (visited on 08/18/2022).

[5] Arthur Drichel et al. "Finding a Phish in a Haystack: A Pipeline for Phishing Classification on Certificate Transparency Logs". In: *Proceedings of the 16th International Conference on Availability, Reliability and Security (ARES)*. Ed. by Delphine Reinhardt and Tilo Müller. Vienna, Austria: ACM, Aug. 2021. ISBN: 978-1-4503-9051-4. DOI: 10.1145/3465481.3470111.

[6] Zakir Durumeric et al. "A Search Engine Backed by Internet-Wide Scanning". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*. Ed. by Ninghui Li and Christopher Kruegel. Denver, CO, USA: ACM, Oct. 2015. ISBN: 978-1-4503-3832-5. DOI: 10.1145/2810103.2813703.

[7] Let's Encrypt. *Certificate Transparency (CT) Logs*. Feb. 25, 2020. URL: https://letsencrypt.org/docs/ct-logs/ (visited on 04/17/2022).

[8] Edona Fasslija, Hassan Ferit Enişer, and Bernd Prünster. "Phish-Hook: Detecting Phishing Certificates Using Certificate Transparency Logs". In: *Proceedings of the 15th International Conference on Security and Privacy in Communication Systems (SecureComm)*. Ed. by Songqing Chen et al. Orlando, FL, USA: Springer, Dec. 2019. ISBN: 978-3-030-37230-9 978-3-030-37231-6. DOI: 10.1007/978-3-030-37231-6_18.

[9] Tobias Fiebig et al. "In rDNS We Trust: Revisiting a Common Data-Source's Reliability". In: *Proceedings of the 13th Passive and Active Measurement (PAM)*. Ed. by Robert Beverly and Georgios Smaragdakis. Vol. 10771. Lecture Notes in Computer Science. Berlin, Germany: Springer, Mar. 2018. ISBN: 978-3-319-54327-7. DOI: 10.1007/978-3-319-76481-8_10.

[10] Tobias Fiebig et al. "Something from Nothing (There): Collecting Global IPv6 Datasets from DNS". In: *Proceedings of the 12th Passive and Active Measurement (PAM)*. Ed. by Dali Kaafar, Steve Uhlig, and Johanna Amann. Vol. 10176. Lecture Notes in Computer Science. Sydney, Australia: Springer, Mar. 2017. ISBN: 978-3-319-54328-4. DOI: 10.1007/978-3-319-54328-4_3.

[11] Josef Gustafsson et al. "A Fist Look at the CT Landscape: Certificate Transparency Logs in Practice". In: *Proceedings of the 12th Passive and Active Measurement (PAM)*. Ed. by Dali Kaafar, Steve Uhlig, and Johanna Amann. Vol. 10176. Lecture Notes in Computer Science. Sydney, Australia: Springer, Mar. 2017. ISBN: 978-3-319-54328-4. DOI: 10.1007/978-3-319-54328-4_7.

[12] Hans Hoogstraaten et al. *Black Tulip: Report of the investigation into the DigiNotar Certificate Authority breach*. Fox-IT, Aug. 2012.

[13] Dali Kaafar, Steve Uhlig, and Johanna Amann, eds. Vol. 10176. Lecture Notes in Computer Science. Sydney, Australia: Springer, Mar. 2017. ISBN: 978-3-319-54328-4.

[14] Brian Kondracki, Johnny So, and Nick Nikiforakis. "Uninvited Guests: Analyzing the Identity and Behavior of Certificate Transparency Bots". In: *Proceedings of the 31st USENIX Security Symposium (USENIX Security)*. Ed. by Kevin Butler and Kurt Thomas. Baltimore, MA, USA: USENIX Association, Aug. 2022.

[15] Nikita Korzhitskii and Niklas Carlsson. "Characterizing the Root Landscape of Certificate Transparency Logs". In: *Proceedings of the 19th Networking Conference (Networking)*. Ed. by Raoui Boutaba and Guy Pujolle. Paris, France: International Federation for Information Processing (IFIP), June 2020. ISBN: 978-3-903176-28-7.

[16] Mohit Kumar. *How Certificate Transparency Monitoring Tool Helped Facebook Early Detect Duplicate SSL Certs*. Apr. 11, 2016. URL: https://thehackernews.com/2016/04/certificate-transparency-monitoring.html (visited on 08/06/2022).

[17] Ben Laurie, Adam Langley, and Emilia Kasper. *Certificate Transparency*. July 1, 2013. DOI: 10.17487/RFC6962. URL: https://www.rfc-editor.org/info/rfc6962 (visited on 03/28/2022).

[18] Ben Laurie et al. *Certificate Transparency Version 2.0*. Nov. 4, 2019. URL: https://datatracker.ietf.org/doc/html/draft-ietf-trans-rfc6962-bis-34 (visited on 03/28/2022).

[19] Fabian Marquardt and Christopher Schmidt. "Don't Stop at the Top: Using Certificate Transparency Logs to Extend Domain Lists for Web Security Studies". In: *Proceedings of the 45th IEEE Conference on Local Computer Networks (LCN)*. Ed. by Karl Andersson. Sydney, NSW, Australia: IEEE, Nov. 2020. DOI: 10.1109/LCN48667.2020.9314793.

[20] Mitre. *CVE-2018-16844*. Sept. 11, 2018. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-16844 (visited on 02/08/2022).

[21] Mitre. *CVE-2018-17189*. Sept. 19, 2018. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-17189 (visited on 02/08/2022).

[22] Mitre. *CVE-2019-8942*. Feb. 19, 2019. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-8942 (visited on 02/09/2022).

[23] Mitre. *CVE-2021-23017*. Jan. 6, 2021. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-23017 (visited on 02/08/2022).

[24] Mitre. *CVE-2021-44790*. Dec. 10, 2021. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-44790 (visited on 02/08/2022).

[25] Johnathan Nightingale. *DigiNotar Removal Follow Up*. Sept. 2, 2011. URL: https://blog.mozilla.org/security/2011/09/02/diginotar-removal-follow-up/ (visited on 08/10/2022).

[26] Devon O'Brien. *Retiring several Google CT logs in Chrome on 01 May 2022*. Apr. 6, 2022. URL: https://groups.google.com/a/chromium.org/g/ct-policy/c/abPZR5silrk (visited on 08/28/2022).

[27] Stijn Pletinckx, Kevin Borgolte, and Tobias Fiebig. "Out of Sight, Out of Mind: Detecting Orphaned Web Pages at Internet-Scale". In: *Proceedings of the 28th ACM SIGSAC Conference on Computer and Communications Security (CCS)*. Ed. by Giovanni Vigna and Elaine Shi. Virtual Event, Republic of Korea: ACM, Nov. 2021. ISBN: 978-1-4503-8454-4. DOI: 10.1145/3460120.3485367.

[28] Ivan Ristić. *Bulletproof TLS and PKI*. Feisty Duck, Jan. 2022. ISBN: 978-1-907117-09-1.

[29] Richard Roberts and Dave Levin. "When Certificate Transparency Is Too Transparent: Analyzing Information Leakage in HTTPS Domain Names". In: *Proceedings of the 18th ACM SIGSAC Workshop on Privacy in the Electronic Society (WPES)*. Ed. by Lorenzo Cavallaro, Johannes Kinder, and Josep Domingo-Ferrer. London, United Kingdom: ACM, Nov. 2019. ISBN: 978-1-4503-6830-8. DOI: 10.1145/3338498.3358655.

[30] Digital in the Round. *What is the Most Popular Web Server Application in 2021?* July 18, 2021. URL: https://digitalintheround.com/what-is-the-most-popular-web-server/ (visited on 02/04/2022).

[31] Sahdowserver. *Hello IPv6 Scanning World!* June 14, 2022. URL: https://www.shadowserver.org/news/hello-ipv6-scanning-world/ (visited on 02/16/2023).

[32] Quirin Scheitle et al. "The Rise of Certificate Transparency and Its Implications on the Internet Ecosystem". In: *Proceedings of the 18th Internet Measurement Conference (IMC)*. Ed. by Ben Y. Zhao and Ethan Katz-Bassett. Boston, MA, USA: ACM, Nov. 2018. DOI: 10.1145/3278532.3278562.

[33] Emily Stark, Joe DeBlasio, and Devon O'Brien. "Certificate Transparency in Google Chrome: Past, Present, and Future". In: *IEEE Security & Privacy* 19.6 (2021), pp. 112–118. DOI: 10.1109/MSEC.2021.3103461.

[34] Emily Stark et al. "Does Certificate Transparency Break the Web? Measuring Adoption and Error Rate". In: *Proceedings of the 40th IEEE Symposium on Security & Privacy (S&P)*. San Francisco, CA, USA: IEEE, May 2019. DOI: 10.1109/SP.2019.00027.

[35] The UNIX and Linux Forums. *urandom(4) [v7 man page]*. Sept. 15, 2017. URL: https://www.unix.com/man-page/v7/4/urandom/ (visited on 04/17/2022).

[36] Benjamin VanderSloot et al. "Towards a Complete View of the Certificate Ecosystem". In: *Proceedings of the 16th Internet Measurement Conference (IMC)*. Ed. by Phillipa Gill and John Heidemann. Santa Clara, CA, USA: ACM, Nov. 2016. ISBN: 978-1-4503-4526-2. DOI: 10.1145/2987443.2987462.

[37] WHATWG. *HTML Living Standard*. Feb. 3, 2022. URL: https://html.spec.whatwg.org/multipage/semantics.html (visited on 02/07/2022).

[38] Wordfence. *Multiple Vulnerabilities Patched in WordPress Download Manager*. July 29, 2021. URL: https://www.wordfence.com/blog/2021/07/wordpress-download-manager-vulnerabilities/ (visited on 02/09/2022).

[39] wpscan. *LayerSlider ¡= 6.2.0 - CSRF / Authenticated Stored XSS & SQL Injection*. May 16, 2017. URL: https://wpscan.com/vulnerability/9e426e65-7373-4934-89c3-42d5c1152a74 (visited on 02/09/2022).

[40] Tarun Yadav and Arvind Rao. "Technical Aspects of Cyber Kill Chain". In: *Proceedings of the 3rd Security in Computing and Communication (SSCC)*. Ed. by Jemal Abawajy et al. Kochi, India: Springer, Aug. 2015. ISBN: 978-3-319-22914-0 978-3-319-22915-7. DOI: 10.1007/978-3-319-22915-7_40.

[41] ZMap. *ZGrab 2.0*. Feb. 8, 2022. URL: https://github.com/zmap/zgrab2 (visited on 02/08/2022).