Abusing Certificate Transparency Logs to Identify Vulnerable Sites

Hanna Sterneling and Robin Ellgren
TDDD17 - Information Security 2019
Linköping University
Linköping, Sweden
hanst665@student.liu.se, robel708@student.liu.se

***Abstract*— Certificate Transparency reformed the internet infrastructure by enabling both users and domain owners to monitor the issuance of certificates. Despite the substantial improvement of security, easy access of newly issued certificates also comes with a risk. Attackers can abuse the information and perform attacks that jeopardize the integrity of targeted systems. Such an attack is evaluated in this paper, specifically targeting Wordpress sites. Experiments confirm that the stream of certificates allows the attacker to harvest vulnerable sites and perform an hijack attack. Even though the number of vulnerable sites is limited and harvesting takes a long time, the consequences of such an attack can cause major damage. We perform this simulated attack on both a certificates extracted from live stream of certificates as well as stored certificates. We also define the attack surface of this attack.**

I. INTRODUCTION

In 2011 an event altered the infrastructure of the entire internet. Due to an attack on one of the Certificate Authorities (CAs), DigiNotar, rouge certificates were issued on domains such as Google.com, Microsoft.com etc. [1]. The attackers could consequently impersonate the websites which the users thought they were visiting. The browser is responsible for determining if a site can be trusted to indeed be the site it claims to be. This decision is based on the certificates that are issued to validate the site. However, as these fraudulent certificates were issued the browsers were not able to detect the mischief. This event, along with some other similar happenings, made it clear that the system, at that time, was not sufficient. In order to manage this problem *Certificate Transparency* (furthermore referred to as CT) evolved, as an initiative from Google.

CT aims to solve the problem of detecting fraudulent certificates in order for a domain owner to take appropriate action. This is achieved by letting CAs add the issued certificates into a publicly auditable log, which is append only. By using modern cryptography the append-only feature is ensured by allowing everyone to monitor the logs for proof of inclusion and also proof of previous included certificates [2].

CT has admittedly improved the security revolving around the trust of websites. It benefits the users as it decreases the risk of fraudulent certificates since domain owners are given the opportunity to keep track on each certificate issued in their name [3]. However, the real time display of issued certificates also poses a problem. The logged certificates contain a lot of information, such as domain (including subdomains), email and expiry date. When such information is made publicly available there is always a risk that the information can be abused to unanticipated means. Thus, CT logs indubitably offers an improvement by quickly making certificates available but that feature also poses a problem. The CT logs effectively creates a never ending public stream of new domains. With new domains there is a stint possibility that the owner of the domain has not yet set up appropriate security measures, leaving a window for an attacker where the attack-surface is sufficiently large.

More specifically, such a vulnerability origins from the usage of Content Management Systems (CMS). WordPress and Drupal are examples of CMS:es and they require an initial installation after uploading the files to the server. Before the initial installation is completed, there is a window for an attacker to hijack the server with arbitrary code execution privileges by accessing the unauthorized installation page. This type of attack has been described by journalist Hanno Böck and he claimed to have found 5000 vulnerable sites under the course of three months [4][5].

This paper aims to verify Böck's experiment, as well as investigate the window in which the site is in a vulnerable state. To complete this task the following questions need to be considered:

- Are there any existing and available tools which can be used to extract information in CT logs? If not, what would be required in order to build a tool?
- Are there any CT logs in particular that are more interesting than others?
- Are Hanno Böck's results viable and can they be reproduced?
- For how long period of time are the detected sites in a vulnerable state?
- Given the time distribution of detected sites in a vulnerable state, can any conclusions be drawn with respect to time delay of CT log appending?

II. BACKGROUND

A. Certificate Transparency

1) Certificates: As briefly described in Section I, certificates are at the center of this paper. The idea is that some trusted node, often referred to as a Certificate Authority (CA), issues an electronic document containing a client's public key and then signs it with its own private key. When

a user visits a web page the certificate is presented. From this certificate the user can extract the site's public key, given that it trusts the issuing CA. If it does not, it can simply ask the CA for it's certificate and repeat the process. This way a chain of trust is created until either the user trusts a CA or if a root CA is reached. Root CAs are often hard-coded into the user's browsers and thus implicitly trusted by using the service of the browser.

2) Certificate Transparency motivation: As pointed out in Section II-A.1, the whole certificate infrastructure depends on a chain of trust leading up to some root CA. Mentioned in Section I, there was an issue with the root CA DigiNotar in 2011, where there were a large amount of fraudulent certificates issued. Obviously a problem arose when it became clear that not even a root CA can be trusted. Enter Certificate Transparency.

B. Certificate Transparency in practice

The concept of Certificate Transparency (CT) originates from the idea that if domain owners can audit their domain for issued certificates they will immediately notice faulty issued certificates and can thus take appropriate action (usually contacting the CA in question and ask for a certificate revocation). Thus CT references to the task of publicly appending all issued certificates (hence the word transparency). Another advantage with CT, besides domain owners auditing their domains, is that CAs also can audit other CAs. This way they can develop systems that take into consideration the existence of certificates from other CAs.

In 2013, a RFC was released specifying the details about CT [6]. As of the time of writing, the RFC is still in experimental mode but in practice it is already widely implemented. Notable events include:

- Since 30th of April 2018, logging of certificates with CT is a requirement for Google Chrome. If a certificate is not logged, the user will be shown a warning that the site is not secure [7].
- Since October 15 2018, logging of certificates with CT is a requirement for the browser Safari. If a certificate is not logged, a TLS connection will be refused [8].
- As for Firefox it was announced in December 2014 that CT were to be implemented. However, since then, to the best of our knowledge, no more announcements from Mozilla regarding CT has been communicated [9].

C. Performing an attack

The content of this section is based on Hanno Böck's presentation at DEFCON25 [4] and verified by the authors.

Given that a vulnerable site has been identified, there is a simple process of gaining full shell-access to the site. The process is also fully automatable and hard to detect for the actual site owner.

Firstly, the attacker goes to the vulnerable site and gets greeted with the Wordpress installation page. Noteworthy is that this site utilizes no authentication.

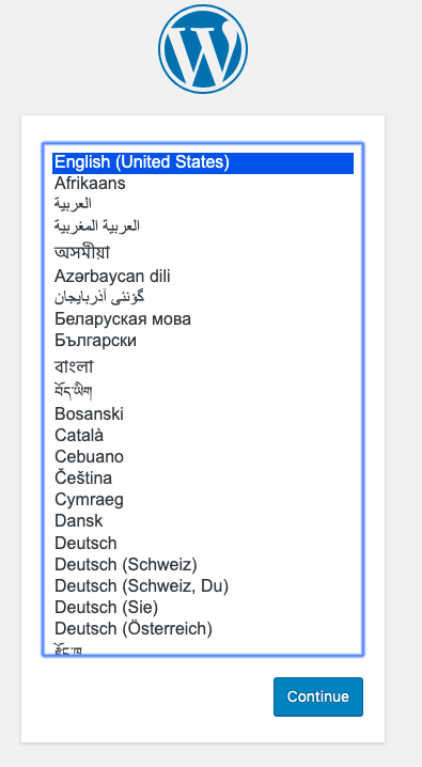

Fig. 1. Screenshot of the Wordpress installation page

Secondly, the attacker follow the simple steps of the installation. Wordpress requires access to a database but has no limits on the database being located on the same server or domain as the site itself. This opens up for the usage of external databases. That means that any database service will suffice, for example *FreeMySQLHosting*. With the database configured the attacker now has access to the Wordpress Dashboard with administrator privileges. Given those privileges, the attacker can upload a plugin containing some PHP code that will execute any terminal command on the server. Suggestively using the *system* function.

```
<?php
if (array_key_exists('cmd', $_GET)) {
        system($_GET['cmd']);
}
?>
```

Thirdly, the attacker navigates to the uploaded plugin and can start executing commands. Navigation is as simple as a GET request, for example:

```
http://<hostname>/wp-content/
plugins/<name of plugin>/shell.php
```

The attacker now has shell access to the host, as illustrated in Figure 2.

Fig. 2. Screenshot of shell access from the web.

Fourth, the attacker wants to hide its traces as well as possible, in order to ensure continuous access. Removing the file *wp-config.php* resets the installation page to the one in Figure 1 and the actual domain owner will not be aware of the intrusion. Then, it is simply a task of writing a good enough description in the plugin to make sure the user does not notice that the plugin should not reside there. If convincing enough, continuous access is ensured.

As Hanno Böck realized the extent of the problem, he reached out to several developers working with CMS:es. However, only Joomla has adapted its installation process to handle this issue.

III. METHOD

In order to answer the questions described in Section I we had to focus on several aspects. Tools, which could ease the process of extracting certificate information, were researched. To validate the experiment performed by Hanno Böck (Section II-C) and also evaluate the time span in which a site generally remains vulnerable, we created our own implementation, which is explained in Section III-B. However, since the experiment performed by Hanno Böck is evidently intrusive, we will begin by describing our experiment from an ethical perspective.

A. Ethics

As explained in Böck's article, he managed to install plugins and in that way exploit the vulnerability. To avoid any possibility of intrusion, such an installation was never attempted in these experiments. However, we did do an attack on *localhost*. For reference, see Section II-C. Instead of performing the attack we simply identified the vulnerable sites and wrote to file. The file containing the vulnerable sites is used to investigate the time period in which these sites remain vulnerable. The file will not be made available to persons outside of the project. Hence, the experiments will not involve any actual attack, only a detection of vulnerable sites.

B. Implementation

The aim of the implementation is to detect vulnerable sites from already existing logs and from a live stream of new certificates. A vulnerable site in this context is one that is in the installation phase of the CMS WordPress and the configurations have not yet been completed. Böck's experiments contained more CMS:es but given the time restraints we decided to focus on WordPress as it was the most common denominator of vulnerable sites in his results. Also, WordPress have not yet taken any action to counteract this vulnerability.

The conducted experiments were divided into three parts:

1) detecting vulnerable sites from certificates found in existing logs,
2) detecting vulnerable sites from certificates issued in real time,
3) investigating the time window in which most of the vulnerabilities where removed.

The primary step is the extraction of information from certificates. To ease this process we used two available tools. *Certificate Transparency Tools* offer the possibility to retrieve decoded information from certificates in a given CT log [10]. We used the logs approved by Chrome as a source [11], however, it was not possible to process all these logs due to too much data. To limit the data set we focused on GoogleArgon as it is one of the biggest logs. A subset of data was gathered from the logs with expiration date 2017, 2019 and 2021. In the interest of examining the HTML in order to determine if the site is vulnerable, the only information needed to be stored are the URLs.

For live stream extraction of certificates, CaliDog provides a tool called *CertStream* [12]. CertStream monitors the *Certificate Transparency Log network*(CT log network) [2] for real-time issuance of certificates. In the same manner as for Certificate Transparency Tools, the certificates are parsed, decoded and supplied to the user.

The next phase was to establish whether or not the URL lead to a vulnerable site. The analysis was simply carried out by visiting each URL, extracting the HTML and attempting to identify components that are unique to the WordPress configuration page. As these sites where detected, the URL was written to file. Figure 3 illustrates how all certificates within the data set was extracted first, followed by a check. The process differed slightly with the live stream, as shown in Figure 4. The vulnerability check was carried out immediately after detecting a new certificate. At this point the implementation satisfied Part 1 and 2 of the fractioned experiments.

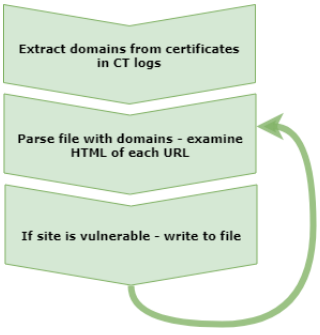

Fig. 3. Flow chart for certificates extracted from CT logs.

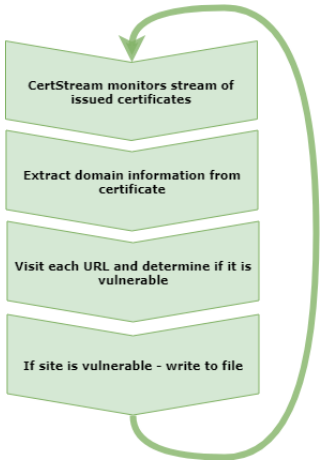

Fig. 4. Flow chart for certificates extracted from live stream.

For Part 3, the implementation for the live stream had to be augmented. As the time of issuance, as well as time of completion of the installation, had to be known, this part of the experiment could only be conducted on the live stream. To determine the window size of the vulnerability we used timestamps. When a URL was deemed vulnerable a timestamp was appended to the saved information. To estimate when the window closes we revisited each site, which was still considered vulnerable, every 15 minutes. If the configuration was completed a final timestamp was added to the file, indicating that the URL was no longer vulnerable. Figure 5 demonstrates the process.

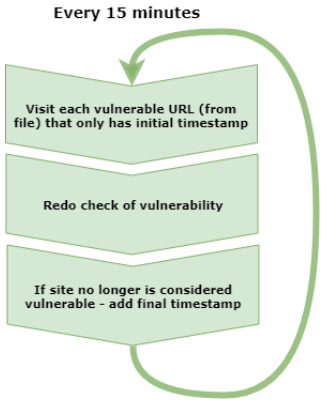

Fig. 5. Flow chart of timestamping sites deemed vulnerable.

IV. RESULTS

The results are divided into three sections to represent the different parts of the experiments, established in Section III. The experiments were carried out over a period of four weeks.

A. GoogleArgon Logs

The log sizes for 2017 and 2019 are ranging from 7 million to over 400 million. Given the time restriction of this project we could only extract a subset and thus settled on extracting the first four million certificates. Table I contains the actual number of certificates at the time of writing found in each GoogleArgon log. Table II contains the actual number of searched certificates for each log as well as the number of vulnerable sites found.

TABLE I
NUMBER OF CERTIFICATES FOUND IN GOOGLEARGON LOGS

Log	Actual size of Data Set
2017	7,842,537
2019	466,000,000
2021	1.015.021

TABLE II
NUMBER OF VULNERABLE SITES FOUND IN GOOGLEARGON LOGS

Log	Size of Data Set	Vulnerable Sites
2017	4,000,000	2
2019	812,758	1
2021	330,112	0

Notable regarding Table II is that for 2017 we aimed at performing an attack analysis on four million certificates but only managed 812.785. For 2021 we set out to perform analysis on the whole log (that is, 1.015.021 certificates) but only managed roughly 300.000. We did not meet our goals in this aspect due to an underestimation of the time the task would take. We did not realize this until well into the project and could potentially have mitigated the skew results by running the task parallel for each log.

B. CertStream

Table III contains the result of the experiment carried out in a period of four weeks. The results indicate that one would find one vulnerable site per 3425 issued certificates.

TABLE III
NUMBER OF VULNERABLE SITES FOUND DURING A FOUR WEEK PERIOD OF LISTENING TO THE LIVE STREAM OF CERTIFICATES

Stream	Size of Data Set	Unique Sites	Percentage
CertStream	709,050	207	0.03 %

C. Window of vulnerability

Figure 6 illustrates the time window in which sites remain vulnerable, as defined in Section III-B. The bars represent the number of sites that completed the configuration within the time span.

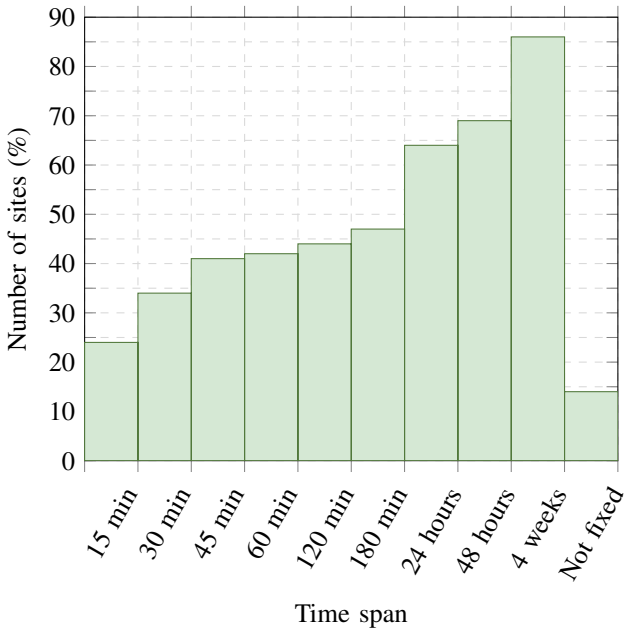

Fig. 6. Each bar represent the accumulated number of sites that closed the vulnerability window within that time frame.

Figure 7 illustrates the cumulative distribution function of the vulnerability window. The increase at the end of the curve represents all those sites that were not fixed during our trial period. To give a better understanding of the first critical hours, where the 50% limit is reached, Figure 8 illustrates the first six hours.

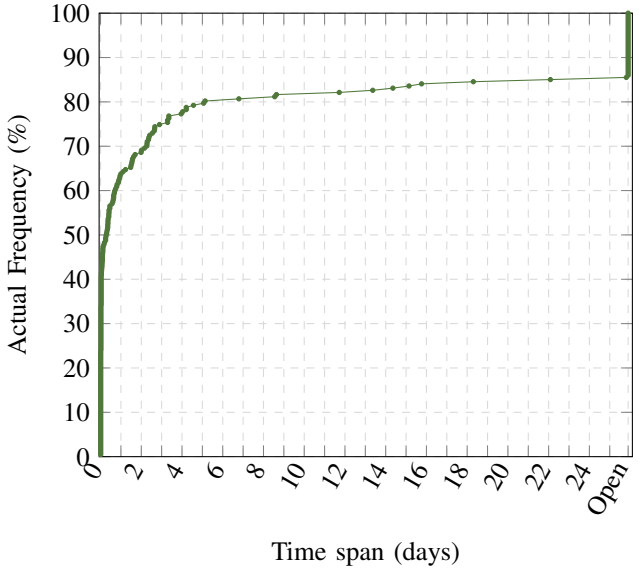

Fig. 7. CDF: Actual frequency of developers completing the configuration

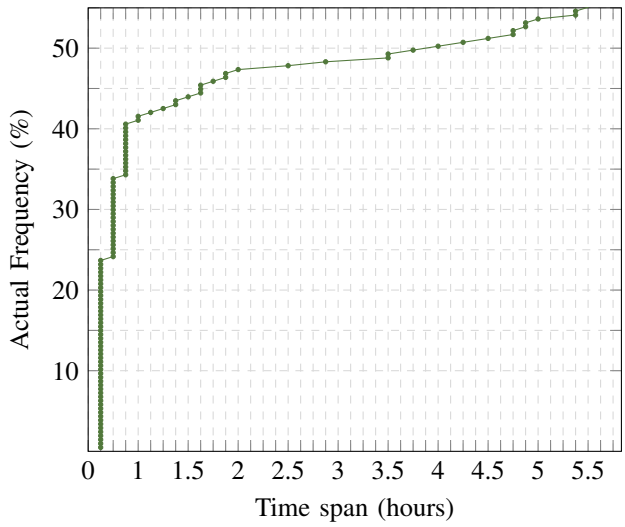

Fig. 8. Figure 7 focused on the first 6 hours

V. ANALYSIS AND DISCUSSION

The results in Section IV-A, especially Table II, clearly show that the attack can not be carried out on the vast majority of sites already present in the CT logs. That is, no new inclusions from the livestream. In fact, a very small portion of all sites registered in CT logs are actually vulnerable. A pattern we were hoping to confirm using the GoogleArgon logs is that the amount of vulnerable sites decreases as time elapses. It would be logical since the more time that has passed more site owners are able to finish the configuration. Unfortunately, we did not collect enough data to make that claim. As stated in Section IV, this was due to an underestimation of how long time analysis on the GoogleArgon logs would take. The consequences of this is that no real claims can be made from the analysis of the GoogleArgon logs other than that only a few vulnerable sites resides there. This means that an attacker can not perform an effective attack on already published certificates. Also, it might be considered alarming that vulnerabilities remain exploitable for years in some sites.

Proceeding, assuming that the amount of vulnerable sites decrease over time, determining the time distribution of how sites addresses the issue seemed like the next logical step. Doing this on GoogleArgon logs does not make sense since there is no way for us to determine when a site got fixed. Therefore we restricted this part of the experiments to the livestream. As seen in Figure 6 around 24% is fixed within the first 15 minutes of publication. Considering that an additional 10%, 7% and 1% is fixed within 15-30, 30-45 and 45-60 minutes respectively a significant 42% part of the attack surface is reduced within the first hour. Thus, a potential attacker has to be relatively fast to ensure maximal effect from the attack. Looking at Figure 7 we can see that after the first hour there is still 58% vulnerable sites.

The number 58% is especially significant because even if there were to be a delay of e.g. one hour before appending

the certificate into the log, it would still be a considerable amount of sites that would remain possible targets. With enough time, an attacker could potentially take over thousands of servers. Beside the obvious consequences, stealing of sensitive information and destruction, this means that an attacker could also build a botnet. With such a wide range of IP addresses and domains the imaginary botnet could for example be the source of a successful spam campaign.

Mentioned in Section I, the German researcher Hanno Böck claims to have found 5000 vulnerable sites under the course of three months. Looking at the results regarding this presented in Table III, we can confirm that the attack is viable and definitely a serious threat. We are however unable to verify the magnitude of his results. Our results show that during the course of three months we would have found approximately 621 vulnerable sites. This is significantly lower and only about 1/8 of what Böck claims. One source of error is that Hanno Böck utilized bash scripting while we used Java, which might impact the speed of visiting sites. It is however hard to make a direct comparison since Hanno Böck has not spelled out the number of visited sites.

VI. CONCLUSIONS

From the results of this project and the analysis of them we have drawn the following conclusions.

- By using a combination of openly available software and self written software we were successful in replicating Hanno Böck's attack.
- We conclude that the attack is viable and a serious threat, which can have major consequences.
- We were unable to confirm the possible magnitude of the attack, as claimed by Hanno Böck. Our results show that there are significantly less vulnerable sites than claimed.
- We were able to determine that the amount of vulnerable sites decreases over time and that the most critical timeframe, from an attackers point of view, is the first hour were around 42% of the vulnerable sites are fixed.

A. Future work

For future work we recognize that determining the potential damage of this attack is important.

1) Determining potential damage: Our project does not take into consideration which type of certificate that is used on the vulnerable sites. Using this information we believe that interesting conclusions could be drawn. For example, one hypothesis is that sites with EV-certificates might be more likely to hold sensitive information, or many registered users, which would consequently result in a higher potential damage. Verifying this with a study and then investigating the amount of EV or OV certificates with this vulnerability could be used to help determining the potential damage this attack can cause. Determining potential damage is also very important with respect to any discussions regarding CT inclusion delay. Furthermore, more work is required on the current logs and more specifically how many vulnerable sites they contain.

REFERENCES

[1] Josephine Wolff. How a 2011 hack you've never heard of changed the internet's infrastructure. https://slate.com/technology/2016/12/how-the-2011-hack-of-diginotar-changed-the-internets-infrastructure.html, 2016. [Online; accessed 8-April-2019].

[2] Certificate-Transparency.org. What is certificate transparency? https://www.certificate-transparency.org/what-is-ct. [Online; accessed 8-April-2019].

[3] Certificate-Transparency.org. Benefits and advantages. https://www.certificate-transparency.org/benefits. [Online; accessed 8-April-2019].

[4] Hanno Bock. Certificate transparency - hacking web applications before they are installed. https://www.golem.de/news/certificate-transparency-hacking-web-applications-before-they-are-installed-1707-129172.html. [Online; accessed 8-April-2019].

[5] Hanno Böck. Ct grab. https://github.com/hannob/ctgrab. [Online; accessed 6-May-2019].

[6] A. Langley B. Laurie and E. Kasper. Rfc 6962 - certificate transparency. https://tools.ietf.org/html/rfc6962. [Online; accessed 8-April-2019].

[7] Google / Chromium. Certificate transparency. https://chromium.googlesource.com/chromium/src/+/master/net/docs/certificate-transparency.md. [Online; accessed 8-April-2019].

[8] Inc. Apple. Apple's certificate transparency policy. https://support.apple.com/en-us/HT205280. [Online; accessed 10-April-2019].

[9] Mozilla. Pki:ct. https://wiki.mozilla.org/PKI:CT. [Online; accessed 10-April-2019].

[10] Erik Tews. Certificate transparency tools @ github. https://github.com/eriktews/certificate-transparency-tools. [Online; accessed 8-April-2019].

[11] Certificate-Transparency.org. Known logs. http://www.certificate-transparency.org/known-logs. [Online; accessed 8-April-2019].

[12] CaliDog. Github - certstream. https://certstream.calidog.io/. [Online; accessed 8-April-2019].