Whom You Gonna Trust? A Longitudinal Study on TLS Notary Services

Georg Merzdovnik[1], Klaus Falb[2], Martin Schmiedecker[1], Artemios G. Voyiatzis[1,2], and Edgar Weippl[1,2]

[1] SBA Research, Vienna, Austria
{gmerzdovnik,mschmiedecker,avoyiatzis}@sba-research.org
[2] TU Wien, Vienna, Austria

Abstract. TLS is currently the most widely-used protocol on the Internet to facilitate secure communications, in particular secure web browsing. TLS relies on X.509 certificates as a major building block to establish a secure communication channel. Certificate Authorities (CAs) are trusted third parties that validate the TLS certificates and establish trust relationships between communication entities. To counter prevalent attack vectors - like compromised CAs issuing fraudulent certificates and active man-in-the-middle (MitM) attacks - TLS *notary services* were proposed as a solution to verify the legitimacy of certificates using alternative communication channels.
In this paper, we are the first to present a long-term study on the operation of TLS notary services. We evaluated the services using active performance measurements over a timespan of one year and discuss the effectiveness of TLS notary services in practice. Based on our findings, we propose the usage of multiple notary services in conjunction with a semi-trusted centralized proxy approach, so as to protect arbitrarily-sized networks on the network level without the need to install any software on the client machines. Lastly, we identify multiple issues that prevent the widespread use of TLS notary services in practice and propose steps to overcome them.

1 Introduction

Secure communication is a key part of today's Internet applications. The majority of online applications, ranging from e-mail to VPN and browsing the web, rely on SSL and TLS[3] to provide secure communication mechansims such as authenticity, confidentiality, and integrity. TLS 1.2 is, at the time of writing, the most recent version [5], with TLS 1.3 currently in the making. Trust in the TLS ecosystem is distributed over software vendors and an underlying public key infrastructure (PKI) composed of various certificate authorities (CAs). To establish a secure connection, a client verifies the signature of a server's certificate.

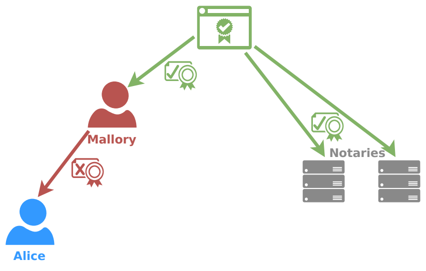
[3] In this paper we use the term "TLS" to refer to all incarnations of SSL and TLS, if not specified otherwise.

If the server's certificate is signed by a trusted certificate authority, the certificate is accepted, otherwise it is rejected. To determine if a CA is to be trusted, the client relies on a so called "trust store", i.e., a list of certificate authorities that it can trust. These trust stores are usually shipped with the application or are included in the operating system. If an attacker gets her hands on one of the private keys of one of these certificate authorities, she is able to issue valid (trusted) certificates for arbitrary-named servers, since the signatures can only be validated against the local trust store. This allows for effective Man-in-the-Middle (MitM) attacks against any kind of targets.

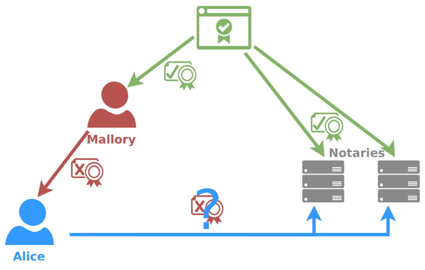
Recent incidents have shown that the subversion of the chain of trust is a viable scenario. Examples include the infamously hacked certificate authorities DigiNotar and Comodo [21], during which their private keys were stolen. Incidents such as the case of Superfish [22] and the Dell eDellroot certificate [31] demonstrate that sometimes even system vendors, like Lenovo or Dell, accidentally introduce vulnerabilities. In these cases, trusted certificate authorities were included in the local trust store of the operating system, which also included the private keys to provide extended functionality, allowing everyone to extract the CA private key and launch unnoticed MitM attacks. For affected users, there is nearly no possibility to distinguish between valid server certificates and those signed by fraudulent CAs, since there are no visible distinction marks and the client's software marks them as trusted.

To solve the problem of multiple valid and trusted certificate chains, several solutions have been proposed recently. These solutions include DANE [15], public key or certificate pinning using HPKP [11], and TLS notary services. The latter are based on the principle of multi-path probing. Figure 1 depicts the usual workflow of such notary services. The idea is to query different "notary" servers if they are presented with the same certificate for a certain communication entity as the client. Therefore, to launch an undetected MitM attack, an attacker would need to intercept as well all the connections to the entity that originate from all the queried notary servers. Since these notary servers are usually spread in different networks around the globe, the risk of an effective, unnoticed MitM attack is highly reduced, even if the certificate is trusted by the local trust store. On the other hand, such a system could reduce the dependability on certificate authorities, since the validation does not have to depend on trusted certificate authorities, but could rely solely on the quorum of a set of notary servers.

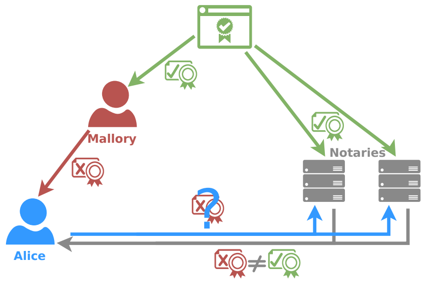
DANE is far from being usable in practice as it relies on DNSSEC which is still not widely deployed. Certificate and public key pinning are still supported only by selected applications (e.g., Chrome, Firefox, and some mobile apps [12,27]). On the other hand, TLS notaries are already implemented as browser extensions, thus being usable in practice. However, there is still no complete study on the long-term usage of notary services and how they react to changes in a real-world setting. In this paper we therefore implement a modular system to



(a) Alice requests a webpage. Mallory intercepts the request and presents a forged certificate.



(b) Alice asks the notaries if they know the certificate.



(c) The notaries' responses tell Alice that the certificate is different from what they have seen.

Fig. 1: The usual flow of a request for certificate notary services.

evaluate notary services in the long term and on a daily basis, independently of the used browsers.

The contributions of this paper are as follows:

– We present a longitudinal study on the effectiveness of three well-known notary services over a one-year period.
– We describe a concept of mapping multiple TLS notaries for transparent end-user protection and an implementation of it as a proxy service.
– We identify problems of combining these services, including lack of widespread adoption and the problem of view inconsistencies.

The rest of the paper is organized as follows: In Section 2, we provide the necessary background on TLS and notary services, as well as the related work. In Section 3, we describe our concept of a proxy notary. In Section 4, we describe our methodology for evaluating the proxy as well as the three TLS notary services independently, whereas our results are described in Section 5. We discuss these results in Section 6. Finally, Section 7 presents the conclusions of this paper and discusses future directions of work.

2 Related Work

Large-Scale TLS Protocol Studies The problems with SSL and trusted certificate authorities have been studied for several years, and several large-scale studies that focused on the TLS ecosystem have been conducted lately. One of the first large-scale studies targeting SSL certificates is the Electronic Frontier Foundation's SSL Observatory [10]. Its dataset includes publicly visible SSL certificates available through IPv4. Holz and Durumeric [17,8] focused on the IPv4-wide analysis of TLS in the context of HTTPS. Mayer et al. [24] and Holz et al. [16] mainly focused on TLS in other application domains, like the e-mail ecosystem. In particular, the recently proposed improvements to port-scanning as well as the open-source release of tools like *zmap* [9] and *masscan* [14] made it easy to collect IPv4-wide information on specific questions. Durumeric et al. also set up a special search engine, Censys [6], which is backed by these Internet-wide scans and allows for deeper analyses. While these studies provide an interesting and valuable view on the TLS ecosystem, they are not designed to provide further information on fraudulently issued certificates.

TLS Certificate Validation Several approaches have been proposed in the literature to mitigate the shortcomings of a central point of trust. One method to provide protection against fraudulent certificates is Certificate Pinning which is already distributed in mobile applications [12,27]. However, recent studies showed that many mobile applications incorrectly implement the validation of TLS certificates [12]. The Internet Engineering Task Force also proposed the Public Key Pinning Extension for HTTP [11]. This allows a web server to set a special header, which tells the browser to only accept a certain certificate or certificates signed by a specific CA, for the specific server and for a specific amount of time.

Other protection mechanisms have been implemented in the form of browser extensions. Soghoian et al. [29] implemented *Certlock*, an extension that is based on the trust-on-first-use policy to bind the CA to the CommonName of a websites certificate. This method is similar to pinning every certificate on first encounter. Winter et al. [35] provide a system that uses an independent Tor circuit for certificates that issued a browser warning. However, this does not protect against valid but yet malicious certificates. Syverson and Boyce also employ Tor for page verification [30], but they do not rely on probing the same server on the same domain; instead, they host the site again on a `.onion` address and use this mirror to compare the keys. Holz et al. [18] implemented CrossBear, a system which employs hunter nodes to track down TLS MitM attacks.

TLS certificate notary services can be used to verify a certificate through multiple paths. Wendlandt et al. [34] proposed *Perspectives*, which is based on multiple servers to observe the state of TLS certificates. *Convergence* [23] builds on the same principles as *Perspectives* and provides further methods for trust management. Bates et al. [2] tried to answer the question of what happens if everyone is using the notary service *Convergence*; Fuchs et al. extended the approach of centralized notary servers and implemented *Laribus* [13], a P2P-based approach on notary services. *Laribus* is based on a social graph, which allows users to build notary groups without the need to rely on a central notary server.

3 Methodology and Measurement Setup

To monitor the effectiveness and behavior of different notary services, we set up an automated crawling environment. Figure 2 gives an overview of the overall design.

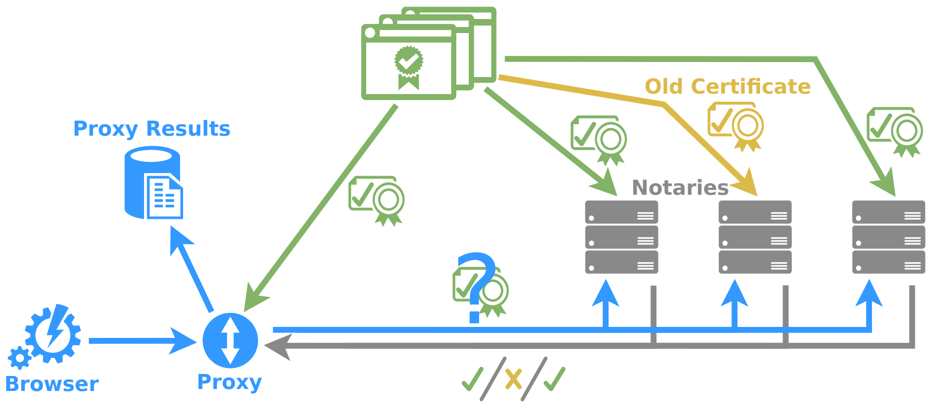


Fig. 2: Overview of our measurement setup.

We implemented the proxy design in `mitmproxy` [4], which allowed us to validate certificates through several extension modules. These extension modules implement interfaces to various notary services, which will be described in Section 3.2. We used this system to collect daily statistics of these implemented extension modules over a one-year period.

3.1 Data Collection System

The data collection system was implemented in such a way that it is extensible, reusable and can furthermore be used by the end users to evaluate their own browsing session. Therefore the overall data collection consists of three components: (1) a web browser, (2) an intercepting proxy to monitor HTTPS sessions, and (3) proxy plugins to query various notary services.

Browser To query the different webpages, we utilized `wget` with the proxy settings pointing to our intercepting proxy. While we used a lightweight, GUI-less browser for our periodic scans, any other full-blown browser could be used as well. This makes the validation proxy described in the next paragraph easier to deploy in combination with other systems. End users can use the proxy to secure or evaluate TLS certificates against various notary services without the need to install separate plugins in their browsers.

Intercepting Proxy To conduct the certificate validation, we implemented an HTTP/HTTPS proxy server in Python 2 using the `mitmproxy` [4] library as a basis. The proxy server acts as an intermediary between the client and the web server. For each encountered HTTPS certificate, the proxy server conducts the certificate validation using the configured notary services.

Proxy Plugins To make the system extensible, the communication with the notary services is implemented as plugins. This makes it easy to extend our system so as to evaluate additional notary services. The proxy in general supports two modes of operation: synchronous and asynchronous. In synchronous mode, the proxy waits for all the responses from the notary services before the original page is passed to the requesting browser. In case of a validation error, this allows to terminate the page load before the page content is rendered to the user.

The second proxy mode asynchronously collects validation information from the notary services and logs them in the file system for later inspection and analysis. In this mode, the page load cannot be interrupted or terminated, since the page is served to the user without waiting for validation responses. For the evaluation, we only look at the results from the asynchronous mode.

3.2 Notary Services

At the time of our initial analysis of the notary services ecosystem, we identified three services that were in use and also had an active and open infrastructure, namely Perspectives, Convergence, and ICSI. We give a short introduction to the inner workings of these systems in the next paragraphs.

Perspectives [3] pioneered the *multi-path probing* approach: The system employs multiple independent servers, called *notaries*, which observe publicly-visible web servers and store data about their certificates. When a client contacts a server using TLS, it queries a number of notaries. The notaries reply with information about which certificate the server in question was using in which time period. Using this information, the client can make a more informed trust decision: Do the notaries see the same certificate as the client?

Convergence [32] was developed by Moxie Marlinspike and builds on the same design principles as Perspectives, but it incorporates other ideas and principles as well. Its central idea is "trust agility", i.e., the users themselves can choose whom to trust and may also revoke their trust. Similarly to Perspectives, Convergence relies on notaries to decide if a certificate is trustworthy or not. However, the decision process is somewhat different. Using a REST web service API, the client sends a request containing the host, port number, and certificate hash to each notary it wishes to query. The server sends one out of five different types of responses, which can be distinguished by the HTTP status code. The possible responses are:

- The notary could verify the certificate.
- The notary could not verify the certificate.
- The notary cannot decide whether to accept or reject the certificate; the client should ignore this notary in its trust decision.
- The client sent a malformed request.
- The server could not perform the request due to an internal error.

This approach makes the implementation of a client rather simple, because the client just has to count the votes collected from the notaries. The protocol is described in more detail in [33]. The user can decide whether decisions are based on majority voting or if an unanimous vote is mandatory in order to accept a certificate.

ICSI Certificate Notary [19] is a service from the University of Berkeley that monitors certificates. In contrast to the two aforementioned services, the ICSI Certificate Notary passively monitors traffic from multiple Internet sites and builds a database of certificates seen in this traffic.

The database can be queried by clients by issuing a DNS query containing the hash of the certificate. The service responds to the client whether it has observed that certificate in the past, and if it could trace this certificate to a valid root certificate through one of the following responses:

1. ICSI has seen this certificate:
   (a) ICSI can establish a chain of trust to a certificate from the Mozilla root store $\rightarrow$ ICSI replies 127.0.0.2 to the request.
   (b) ICSI cannot establish a chain of trust $\rightarrow$ ICSI replies 127.0.0.1
2. ICSI has not seen this certificate $\rightarrow$ ICSI replies with an `NXDOMAIN` error message or an error (such as a time-out) has occurred $\rightarrow$ no reply[4].

Note that ~~(it is not possible to)~~ we do not distinguish between the cases *"a query timed out"* and *"ICSI has not seen this certificate"*, therefore our proxy plugin rejects the certificate in both cases.

[4] We are grateful to Johanna Amann from ICSI for clarifying the difference. The proceedings version of the paper daeso not include this correction.

4 Data Collection

Our data collection involves periodic TLS certificate validation requests to the set of analyzed notary instances for 1,000 web pages. The scans were conducted daily, and each scan involved queries to the three different notary services for each of the encountered certificates. We conducted the evaluation of the validation proxy in two steps: First, we collected a sample set of pages served through HTTPS. Secondly, we conducted daily scans to validate the corresponding TLS certificates against different notary services and analyzed their responses.

4.1 Sample Selection

To select samples, we initially obtained the list of Alexa Top 1,000,000 sites [1] on November 29, 2013. From this list, we then selected the top 1,000 sites that responded to an HTTPS query within 30 seconds. This selection represents the websites that attract most of the visits by users, including pages such as Facebook, Twitter, and Google. Many of the selected websites did respond to HTTPS queries, but with an immediate redirection to a (non-secure) HTTP connection. This means that while they do support HTTPS, many users will probably not use it. However, we still included these sites in the evaluation under the assumption that HTTPS is likely to be used in some parts of the website, like the login pages.

4.2 Periodic Scan

Between January 31, 2014 and January 29, 2015, for a period of one year, the collection was conducted daily. For each run of the scan, the proxy server was started and the previously selected URLs were queried, with the different notary plugins enabled. The data returned by the proxy plugins as well as the collected certificates were stored for further analysis. To get a baseline for comparison, we also queried the URLs without using a proxy server. Thus, in one evaluation run, each site from our data set was queried for a total of four times.

For each pair of URL and validation method, the following measurements were taken:

Verdict: Whether the validation method *accepted* or *rejected* the site's X.509 certificate.

Reason: The reason why a certificate was rejected, if it had been rejected. This metric is specific to each validation method.

Validation Time: The entire time the validation process of a certificate took, including querying the notary server(s) and waiting for a response.

5 Results

In the following, we describe our results and findings from the collected dataset. For each notary service, we analyzed how long it took to answer a validation request and also how long it took to react to certificate changes. Furthermore, we studied the availability of these services over the course of one year.

5.1 Certificate Changes

To analyse the functionality of notary services, it is important to observe actual certificate changes. Figure 3 depicts the number of different certificates per website we encountered during the course of our study. In 80% of the cases, the websites changed at least once their certificate; some 10% of them changed more than 9 times their certificate within the one year that our study was active.

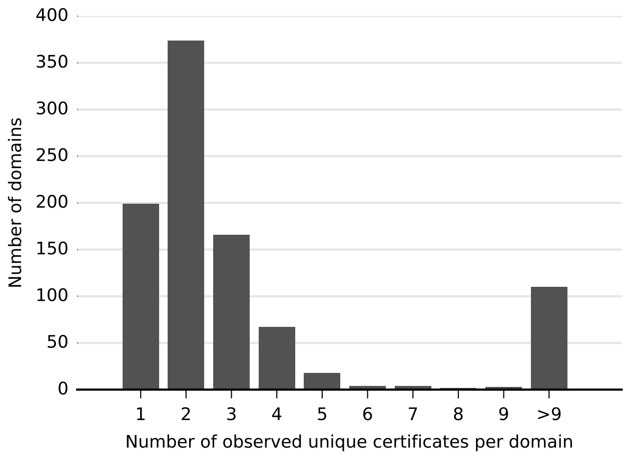


Fig. 3: Number of different certificates observed for each tracked domain

5.2 Validation Time

An important factor concerning notary services from a usability point of view is their response time to validation requests. Therefore, we conducted an analysis of the response time of the various services. With the 1,000 webpages crawled daily for one year, we collected in total more than 350,000 response timing samples per analyzed notary service. Figure 4 summarizes the timing information for the three notary services.

The DNS-based approach of ICSI yields the fastest responses to queries, with the

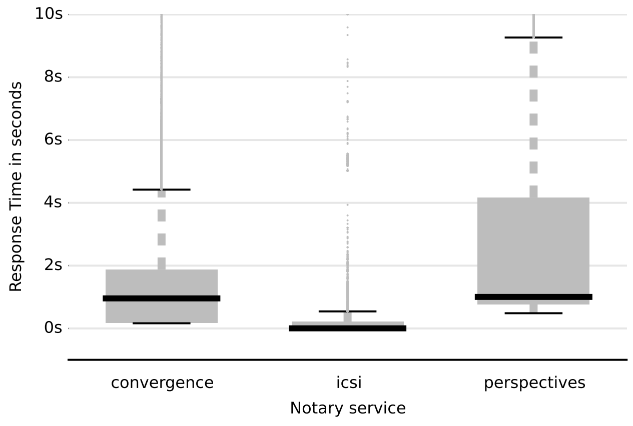


Fig. 4: Response times of notary services to validation requests. Outliers are cut off at 10 seconds.

majority (95%) of answers received in under one second. While about half of the responses for Convergence and Perspectives are also below this mark, response times for these two services have a far higher fluctuation. This can be an issue in the case where the notary services are used to validate certificates before a page is loaded, as it could introduce noticeable page load delays for the users. We note that Convergence usually employs a client cache for fingerprints, in an effort to improve the loading times. We did not implement this caching in our proxy so as to get a comparison of the notary service based on newly-encountered pages.

5.3 Certificate Acceptance Duration

While the response time is certainly important for the general usability in day-to-day browsing, another temporal factor to take into consideration is the time a notary needs to mark new or changed certificates as valid once they are introduced or updated. Figure 5 depicts the time it took the different services to mark a new certificate as valid after it was changed on the server. Since we conducted daily crawls at a fixed time, the resolution of our scan is also on a daily basis. Therefore, a value of zero days means that the certificate was changed by a server as well as validated by a service within this 24-hour time frame. The validations are set in relation to the total amount of certificate changes that we observed during our scanning period. In the case of Convergence we only considered the server we setup ourselves and did not include the official server results, since the latter only responded in error for the majority of our scans.

As depicted in Figure 5, it takes less time to Convergence so as to adopt to newly-changed certificates, with the majority of certificates seen as valid within

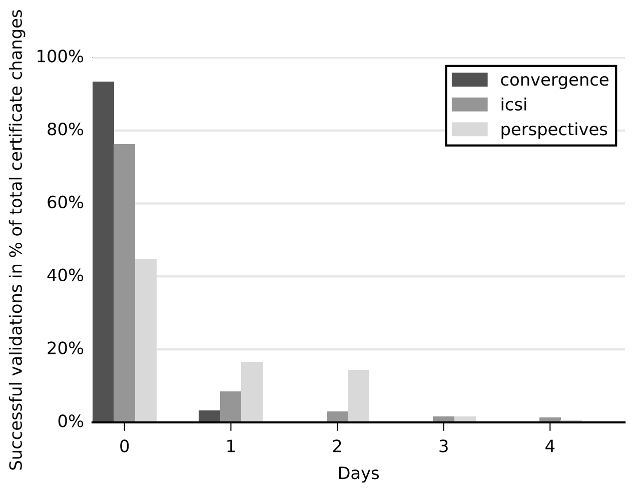


Fig. 5: Time until a newly-seen certificate is marked as validated in percent of the total of observed certificate changes within the period of one year.

the first 24-hour time frame. ICSI is only able to validate about 75% of changed certificates within the same time frame. This fact could be due to the nature of ICSI, which relies on passive information collection, whereas Convergence actively probes servers itself. The relatively low validation rate of Perspectives (45%) can most likely be accredited to the fact that more and more of the servers failed; in the end, it was not possible to reach a quorum on the validity of a certain certificate. Therefore, some of the changed certificates could not be validated successfully anymore. However, even with these limitations in mind, we can still see the general trend that it takes a longer time for Perspectives to successfully validate certificates compared to the other two services. It takes one day for Convergence and at most three for ICSI to fully synchronize.

5.4 Service Availability

Figure 6 provides an overview on the status of certificate validation of the three notary services during the course of one year. The return state for each of the services is given as a percentage for all the collected service responses. It shows the daily average of responses to the 1,000 page request made by the respective crawler.

ICSI was constantly up and running during our scan. We experienced several problems with both Convergence and Perspectives. The analysis of Convergence was based on two servers. The first one was the official server available at *notary.thoughtcrime.org* and the second one was a server we hosted on an Amazon

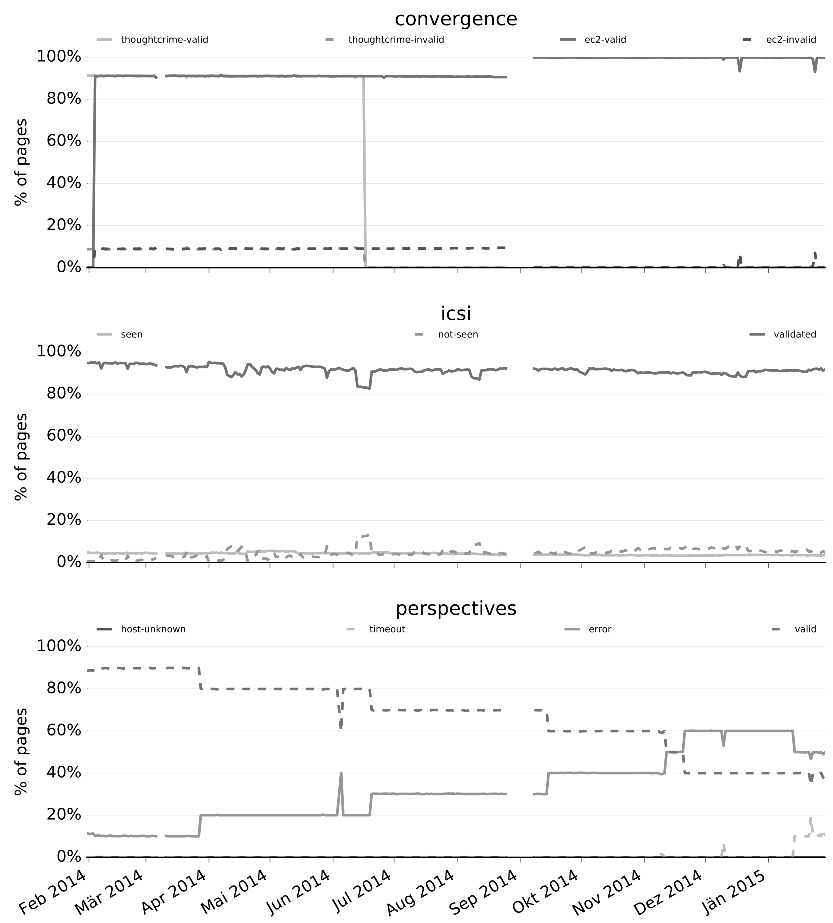


Fig. 6: Timeline of the responses collected from the different services over the course of one year.

EC2 instance. The official server became unresponsive in the middle of June 2014.

We encountered a similar problem with Perspectives, where the initially-available servers one after the other shut down or responded in error. As described earlier, the Perspectives validation of certificates operates with a quorum-based approach, in which at least a certain amount of servers must provide a valid response. Due to the fact that more servers answered with an error state, this requirement was no longer met and therefore, from a certain point in time, all certificates were rejected by the system, even if some of the servers still provided a valid response.

6 Discussion

Analyzing notary services on a longitudinal scale reveals several problems and shortcomings that limit the usability of these services. In the following we discuss the observed limitations and possible future directions for the deployment of notary services.

6.1 Response and Validation Times

One problem with notary services is the delay that these services introduce in page requests. As we described in Section 3.1, there are two approaches to verify a certificate through a notary: synchronous and asynchronous. Both approaches have positive and negative sides. Since the synchronous method waits for all notary responses before actually requesting the page, it can introduce a significant delay (as shown in Section 5.2) in page loading, especially if a notary server times out. On the other hand, the asynchronous method loads the page before it receives all notary responses, therefore leaving a window of exposure before notifying the user that something went wrong.

Another problem is the reaction to legitimate certificate changes, namely how long it takes until a service marks a newly seen certificate as valid. Our study shows that it can take up to several days until a certificate is considered as valid. Until the new certificate is validated and has been seen by all the notary services, it will appear as a MitM attack.

6.2 Adoption and Continuous Operation

For a notary service (or multi-path probing in general) to be useful for actually validating certificates, there are two important factors that need to be met: (i) Services need to be adopted by the users. This means that users have to run their own servers which others can query. For example, if there is only one official server you can query, this defeats the whole concept. (ii) It implies that one has to fully trust this service, which introduces a single point of failure. A single server could just provide wrong answers to the client's queries without a possibility to check these claims, which would be similar to a device-hosted trust store. On the other hand, even if users set up their own servers, the question is how long they can keep them up and running for other clients to use. Therefore, an important factor to consider is that the amount of available servers could fluctuate. The clients need to be informed of failing servers, since this influences the weight of still-running services in the case of majority voting.

Currently it seems that the adoption of these services by users is low. At the time of the writing, the Firefox Add-on for Perspectives has 5,334 users [28] and the plugin for Convergence only 77 users [25]. During our study some of the official servers seem to be discontinued, which does not help to increase the trust in this system. What our insights show is that either the incentive for users to host their own notary services has to be increased or the system itself has to be adapted. One possible adaption is presented by tofu [20], proposing a P2P-based system in which every client is automatically also a host. While this system may be able to solve the problem of service availability, it could still impose further risks that need to be analyzed in the future.

6.3 Privacy

Beside the technical aspects, there are others to be considered. One of them are possible privacy implications. By using a third-party service to validate certificates, it is easy for its server(s) to collect information about the pages a client visited. Therefore it is possible for the server(s) to build a browsing profile of the specific user. One solution to this problem would be for the users to host their own servers. However, this is not always an option and future research should focus on the possibilities to validate certificates without giving away too much information about the client.

While we do not have solutions to these problems (yet), we still believe that notaries are a viable alternative to increase the overall security of TLS. Thus, they should be studied further so as to improve the current limitations.

7 Conclusion

In this work, we presented a longitudinal study on the availability and functionality of different notary services. We conducted daily scans over a period of one year and analyzed the collected data. We explored the ecosystem of notary services and analyzed their behavior on a large scale. To conduct this study, we developed a new proxy-based system to transparently query different notary services for increased protection against MitM attacks and gave an overview on the inner workings of these notary services. Lastly, we discussed the results of our study in the context of the available notary services. We described the problems and pitfalls that can arise by using the existing systems.

Existing notary services have the problem that the initial request for an unknown page can introduce extra latency, since the notary has to query the server for the certificate. With the rise of fast, Internet-wide scanning solutions, there are several projects that analyze the TLS landscape. One of these projects is `scans.io` [26] which hosts a regularly-collected dataset of TLS certificates. `Censys.io` [7] provides a search engine over the scans.io datasets. Future research could evaluate the possibilities to use these data sources either as alternative initial data providers to bootstrap notaries or to wrap the data into a separate notary service.

Acknowledgements

This work has been carried out within the scope of “u’smile”, the Josef Ressel Center for User-Friendly Secure Mobile Environments, funded by the Christian Doppler Gesellschaft, A1 Telekom Austria AG, Drei-Banken-EDV GmbH, LG Nexera Business Solutions AG, NXP Semiconductors Austria GmbH, and Österreichische Staatsdruckerei GmbH. Additionaly the research was funded by the Austrian Research Promotion Agency (FFG) through the Bridge Early Stage grant P846028 (TLSiP) and the COMET K1 program.

References

1. Alexa. Alexa top domains. `http://www.alexa.com/topsites`.
2. A. Bates, J. Pletcher, T. Nichols, B. Hollembaek, and K. R. Butler. Forced perspectives: Evaluating an SSL trust enhancement at scale. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, IMC ’14, pages 503–510, New York, NY, USA, 2014. ACM.
3. CMU. Perspectives project. `http://www.perspectives-project.org/`, 2016.
4. A. Cortesi. mitmproxy. `https://mitmproxy.org/`, Feb. 2016.
5. T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), Aug. 2008. Updated by RFCs 5746, 5878, 6176.
6. Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman. A search engine backed by Internet-wide scanning. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS ’15, pages 542–553, New York, NY, USA, 2015. ACM.
7. Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman. A search engine backed by Internet-wide scanning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 542–553. ACM, 2015.
8. Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman. Analysis of the HTTPS certificate ecosystem. In *Internet Measurement Conference*, 2013.
9. Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-wide scanning and its security applications. In *Usenix Security*, volume 2013, 2013.
10. P. Eckersley and J. Burns. An observatory for the SSLiverse. *Talk at Defcon*, 18, 2010.
11. C. Evans, C. Palmer, and R. Sleevi. Public key pinning extension for http (hpkp). RFC 7469 (Draft), 2015.
12. S. Fahl, M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, and M. Smith. Why eve and mallory love Android: An analysis of Android SSL (in) security. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 50–61. ACM, 2012.
13. K.-P. Fuchs, D. Herrmann, A. Micheloni, and H. Federrath. Laribus: privacy-preserving detection of fake SSL certificates with a social p2p notary network. *EURASIP Journal on Information Security*, 2015(1):1–17, 2015.
14. R. D. Graham. Masscan: Mass ip port scanner. *URL: https://github.com/robertdavidgraham/masscan*, 2014.

15. P. Hoffman and J. Schlyter. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698 (Proposed Standard), Aug. 2012. Updated by RFC 7218.
16. R. Holz, J. Amann, O. Mehani, M. Wachs, and M. A. Kâafar. TLS in the wild: an Internet-wide analysis of TLS-based protocols for electronic communication. *CoRR*, abs/1511.00341, 2015.
17. R. Holz, L. Braun, N. Kammenhuber, and G. Carle. The SSL landscape: a thorough analysis of the x. 509 pki using active and passive measurements. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 427–444. ACM, 2011.
18. R. Holz, T. Riedmaier, N. Kammenhuber, and G. Carle. X.509 forensics: Detecting and localising the SSL/TLS men-in-the-middle. In *Computer Security–ESORICS 2012*, pages 217–234. Springer, 2012.
19. ICSI. Icsi certificate notary. `http://notary.icsi.berkeley.edu/`, 2016.
20. R. Jones. tofu:// - the web security protocol which should have been. `https://gun.io/blog/tofu-web-security/`, Oct.
21. N. Leavitt. Internet security under attack: The undermining of digital certificates. *Computer*, 44(12):17–20, 2011.
22. Lenovo. Superfish vulnerability. `https://support.lenovo.com/at/de/product_security/superfish`, Oct. 2015.
23. M. Marlinspike. SSL and the future of authenticity. *Black Hat USA*, 2011.
24. W. Mayer, A. Zauner, M. Schmiedecker, and M. Huber. No need for black chambers: Testing TLS in the e-mail ecosystem at large. *CoRR*, abs/1510.08646, 2015.
25. Mike Kazantsev. Convergence extra Firefox Add-on. `https://addons.mozilla.org/en-US/firefox/addon/convergence-extra`, Apr. 2016.
26. U. of Michigan. Scans.io - Internet-Wide scan data repository. `https://scans.io/`, Feb. 2016.
27. M. Oltrogge, Y. Acar, S. Dechand, M. Smith, and S. Fahl. To pin or not to pin—helping app developers bullet proof their TLS connections. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 239–254, 2015.
28. perspectives-cmu and D. Schaefer. Perspectives Firefox Add-on. `https://addons.mozilla.org/en-US/firefox/addon/perspectives/`, Apr. 2016.
29. C. Soghoian and S. Stamm. Certified lies: Detecting and defeating government interception attacks against ssl (short paper). In *Financial Cryptography and Data Security*, pages 250–259. Springer, 2012.
30. P. Syverson and G. Boyce. Genuine onion: Simple, fast, flexible, and cheap website authentication. In *Proceedings of the 9th Workshop on Web 2.0 Security and Privacy (W2SP) 2015*, 2015.
31. L. P. Thomas. Response to concerns regarding edellroot certificate. `http://en.community.dell.com/dell-blogs/direct2dell/b/direct2dell/archive/2015/11/23/response-to-concerns-regarding-edellroot-certificate`, 2015.
32. Thoughtcrime Labs. Convergence. `http://www.convergence.io/`, 2016.
33. Thoughtcrime Labs. Convergence notary protocol. `https://github.com/moxie0/Convergence/wiki/Notary-Protocol`, 2016.
34. D. Wendlandt, D. G. Andersen, and A. Perrig. Perspectives: Improving SSH-style host authentication with multi-path probing. In *USENIX 2008 Annual Technical Conference on Annual Technical Conference*, ATC'08, pages 321–334, Berkeley, CA, USA, 2008. USENIX Association.
35. P. Winter, R. Köwer, M. Mulazzani, M. Huber, S. Schrittwieser, S. Lindskog, and E. Weippl. Spoiled onions: Exposing malicious Tor exit relays. In *Privacy Enhancing Technologies*, pages 304–331. Springer, 2014.