Certificate Revocation

Raphael Schmid, Juliane Aulbach*, Patrick Sattler*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: raphael.schmid@tum.de aulbach@net.in.tum.de, sattler@net.in.tum.de

Abstract—
This paper explains the concept of certificate revocation and how it is implemented and enforced in the real world. It explains why certificate revocation is necessary and how there is no agreed-on standard to execute it. It will introduce the actors in certificate revocation and how they fulfill their role in ensuring user security. Then the paper will explain various approaches to optimize certificate revocation and compare them to conclude which one is the most suitable for usage.

Index Terms—**certificate revocation, internet security, revocation methods**

1. Introduction

In the TLS protocol, having the ability to revoke certificates when they become invalid before their validity period ends is crucial. However, there is no agreed-upon standard in the industry as to how to handle revocation. It results in most browser clients not adequately checking for revocation when connecting with a server. It can allow attacks like "man-in-the-middle attacks". This paper introduces the necessary background information to understand certificate revocation in chapter 2. Then it explains how certificate revocation works and its most used methods in chapter 3. In chapter 4, the behavior of browsers is explored and what their different shortcomings are when it comes to certificate revocation. Then in Chapter 5, it discusses which problems can arise from the server, which must request a certificate revocation if a certificate gets compromised. Chapter 6 explores which different classifications there are for handling certificate revocation and then introduces some of the most recent approaches, concluding with a comparison of said approaches. Chapter 7 then evaluates the findings of the previous chapters and Chapter 8 finishes with a conclusion of the paper.

2. Background

The secure exchange of data on the internet is a big concern for many people nowadays. A widely used standard to ensure this security is the TLS protocol. TLS stands for Transport Layer Security. It utilizes a twofold system. One part of this system is encryption to ensure confidentiality and integrity of the data. The other part is to verify the identity of a remote party using digital certificates. [1], [2]
These digital certificates will be the main focus of this paper. Here is a real-world example of how they work. Say a client, e.g. a web browser, wants to connect to a server. To verify its identity to the client, the server sends a digital certificate to the client, which is specifically issued for the domain of the server. This digital certificate identifies the server to the client. But with just that, neither the validity of the server nor the certificate is guaranteed. This is attained using a so-called *chain of trust*. [1], [2]
The chain starts with a certificate authority (CA) which presents a *root certificate*. A CA can issue a certificate for any domain. The issued certificate gets signed by the CA ensuring that the certificate can always be traced back to the CA. If this certificate now can also issue and sign certificates itself, it is called an *intermediate certificate*. A certificate that is not able to sign another certificate is called a *leaf certificate*. Leaf certificates are used by most websites. *Root certificates* are assumed to be trusted by the client. When a client now wants to verify a certificate it has to follow this chain, starting with the root certificate, traversing zero or more intermediate certificates until it reaches the leaf certificate. The client has to verify every signature of every certificate it traverses. [1], [2] This mechanism is visualized in figure 1 below.

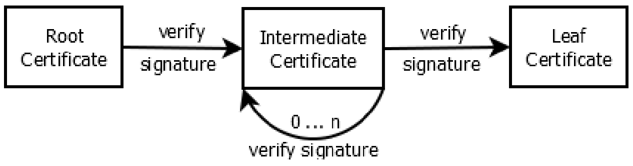

Figure 1: Chain Of Trust

3. Certificate Revocation

Certificates are not valid indefinitely. Every certificate has a validity period [1]. But, there can be occasions where the certificate is not valid anymore before the said validity period is over. Examples for this could be that the private key of a certificate became compromised, was generated with a weak algorithm, or the erroneous issuance of a certificate. [3] If that happens, the certificate must get revoked, advertising to entities checking the certificate that it is no longer valid [1]. Revoking a certificate, in this case, is important. If a certificate got compromised but not revoked, it could open some serious vulnerabilities. For example, an attacker impersonating the identity of the webserver or eavesdropping on private communication. [2]
There are three key actors in certificate revocation:

- Web server / Server administrator [1], [2]
- Certificate authority (CA) [1], [2]
- Client (e.g. web browser) [1], [2]

Here is how a certificate revocation would play out in real life: When a certificate needs to get revoked, first, it is the responsibility of the web administrator to send a revocation request to the CA which signed the certificate. The CA then must sign a statement that the certificate got revoked and is responsible for disseminating this information on the internet. [1], [2] A client that wants to connect to the server managed by the web administrator then is responsible for checking the status of a certificate used by a server to establish a secure connection. There are two methods most often used by CAs to disseminate revocation statuses of certificates online. CRLs and OCSP. [1]

3.1. CRL

A Certificate Revocation List (CRL) is a file that contains a list of all revoked certificates of a CA. Every CRL has an expiration date and must get updated and published periodically by the CA. To check if a certificate got revoked, the client must download the CRL from the CA and then check for the certificate inside the downloaded CRL. The client can cache the CRL until its validity expires. But still, making the client download a whole list of revoked certificates imposes a burden on the client and can add to latency when loading a web page. It is especially true for bigger CRLs. Apple, for example, had a CRL with 2.6 million revoked certificates which had 76 MB in size. [1]

3.2. OCSP

OCSP, Online Certificate Status Protocol, aims to address this problem of high overhead. It reduces the overhead compared to the CRLs by making the client query for the revocation status of just a single certificate. Using OCSP, the client generates an HTTP request to check the serial number of a certificate. The CA then sends the information to the client with a signed response. But this approach introduces a new problem. It contains a privacy issue, exposing the client's browsing behavior to the CA. Also, it still puts some burden on the client and adds to latency. [1]

4. Browsers

In a real-world scenario, the client connecting to a server is typically a browser. This section will explain the behavior and shortcomings of browsers in certificate revocation.
Browsers behave differently on varying devices. There is no mobile browser that checks for certificate revocation. The reason for this lack of security is to decrease the cost, regarding latency and power, for the mobile devices. [1] But, like already discussed, not checking for the revocation status of certificates opens up some serious security vulnerabilities, which is alarming considering that 55.56% of web traffic is generated by mobile phones [4].
When observing the behavior of desktop browsers, there is no common behavior as to how to handle certificate revocation. No browser acts the same. It is also worth noting that even for the same browser, the behavior can change depending on the platform and the type of certificate. Since it is exceeding the scope of this paper to consider all these different factors, this paper will only look at Google Chrome and Mozilla Firefox as examples of shortcomings of browsers handling certificate revocation. Also, only the behavior on Microsoft Windows will be covered, since it is the most used operating system. [5] ([1] takes an in-depth analysis on this matter.)
Google Chrome does not check CRLs for certificate information. Chrome uses a customized CRL to look up revocations called the *CRLSet*. The CRLSet is a 250 kB big, pre-populated list of certificates that get put together by Google by crawling a pre-set list of CRLs. It is not clear what Google's criteria are for these CRLs. This list of CRLs is already just a small subset of the CRLs existing on the internet but Google then also applies a list of rules to drop more revocation data. Which results in Google covering only 75.6% of the revoked certificates that they consider. All of this leads to Google covering only 0.35% of all revoked certificates online in their CRLSet. [1] [2]
Mozilla Firefox uses NSS for certificate verification. "Network Security Services (NSS) is a set of libraries designed to support cross-platform development of security-enabled client and server applications." [6] Firefox disabled CRL checking and only uses OCSP requests. Extended Validation (EV) certificates are special certificates containing additional information offering a more careful verification. Firefox differentiates between EV and non-EV certificates, checking only the leaf-certificates for non-EV certificates. This is a behavior displayed by many browsers. [1] Like Googles' CRLSet, Firefox uses OneCRL, a list of pre-stored certificates [7], [8].

5. Servers

Server administrators are responsible for issuing revocation requests to their respective CA. A human component in the process of certificate revocation opens some security vulnerabilities. This showed in the aftermath of the Heartbleed event in 2014, a "vulnerability in OpenSSL's implementation of the Heartbeat Extension" which caused a mass revocation of digital certificates. [9] It exposed that even after 3 weeks 10% of vulnerable websites still had not addressed the issue. And also showed the number of revocations went down on the weekend, probably because there was no server administrator monitoring the security of the server during that time. And finally, of the certificates that were reissued because of Heartbleed 4.1% of certificates were reissued with the same private key. [9]
This raises the question of whether an important task like revoking certificates, should be done manually and be prone to human error.

6. Methods of revocation validation

There are three different types of approaches to revocation validation.
When utilizing a **pull-based** approach, the client requests the revocation status of a certificate when needed [10].

CRLs and OCSP classify as pull-based approaches.
Second is the **push-based** approach, where the client periodically downloads revocation information. This approach does not reveal the client's traffic patterns, while most of the pull-based approaches do. [8]
Last is the **network-assisted** approach. The idea is to change the ecosystem of TLS in a way that makes it unnecessary for the client to request the revocation status of a certificate. [1], [8], [10]

6.1. CRLite

CRLite is a push-based approach. The idea is to push all certificate revocations to the browser periodically utilizing a two-part system. [11] The first part happens on a server, where all known TLS certificates are getting crawled on the web. All of the revoked certificates get hashed into a bit-vector. If the bit assigned to a certificate is 1, the certificate got revoked. To avoid the risk of false positives, confusing a non-revoked certificate with a revoked one, this mechanism gets repeated with a set of all revoked certificates and all non-revoked certificates until no false positives are left. The filter used is called a bloom filter and this technique of cascading down many filters to eliminate false positives is a bloom filter cascade. Avoiding false positives is crucial to allow the client to hard-fail. [11] The second part is on the client-side, downloading the filters and using them to check for the revocation of observed certificates. [11] CRLite aims to maximize the efficiency of checking for revoked certificates. Only 10 MB are needed for roughly 30 million certificates. Once downloaded they can be updated on a daily basis averaging about 580 kB. CRLite can be deployed by simply installing a plug-in on the browser. [11] However, when CRLite was proposed the bloom filter cascade was only 10 MB. But already a year after the proposal the filter used up 18.1 MB of space [10]. This is because between January 2017 and January 2020 the number of live certificates went up from 30 million to over 434 million. This is because of services like Let's Encrypt which enables automatic issuing for certificates, as well as efforts to normalize using only encrypted web traffic. [8]

6.2. Let's Revoke

Let's Revoke is an approach based on CRLite. But compared to CRLite, Let's Revoke needs 28% of network bandwidth. It utilizes a push-based model with a focus on minimizing network bandwidth consumption while maintaining a global revocation coverage. [8] To achieve this Let's Revoke invented a method using so-called Certificate Revocation Vectors (CRV), Revocation Numbers (RN), and Revocation IDs (RID). CRVs are dynamically-sized bit vectors. Each bit in the vector represents the revocation status of one specific certificate. The bits in the vector are mapped to their respective certificates with the RN. To limit the size of the vectors and ease the use, they are separated by date of expiration. RIDs are then used to identify which CRV a certificate belongs to. To deploy Let's Revoke, it is necessary to make adjustments on the client as well as on the CA side. Incremental deployment is also possible. [8]

6.3. CRT

A certificate revocation table (CRT) is a pull-based revocation approach. To make it work, a server maintaining a certificate working set is needed. The working set gets updated periodically by querying OCSP responders or CRL endpoints. The certificate information can be accessed by the clients by either downloading a file or an on-demand API. [10] CRT maintains a cache containing the revocation status of certificates with a high probability of usage. This is expected to make it more difficult for attackers to perform an attack and allows the client to hard-fail. CRTs can contain revoked and non-revoked certificates. When only used for revoked certificates it does not use a lot of bandwidth. The required bandwidth for a CRT is easily scalable, also over the next years, since it is only directly influenced by the number of certificates used by the client. Even during a mass revocation event, it would remain steady since the revocation status of every certificate is already in the working set of the CRT no matter if it is revoked or not. CRT allows being updated according to the needs of the consumer. It allows for the privacy of the client. A CRT would need to be deployed on the server by an administrator. CRT is still new and there are plans for improvements, which can be read about in [10].

6.4. OCSP Must-Staple

OCSP imposes a burden on the client and exposes the browsing behavior of the client to the CA responsible for issuing the certificate to the server. To combat these problems, OCSP Stapling got introduced. Using OCSP Stapling a server must periodically query an OCSP request to the CA and cache this signed response, proving the validity of the certificate. When a client then wants to establish a connection to the server, the server must send the signed response stapled with the certificate to the client during the TLS handshake. [3]
OCSP Stapling solves the initial problems of OCSP. However, the clients can still choose to continue connecting to the server if the OCSP response is not provided. Connecting without a signed response is called soft-failing. OCSP Must-Staple was invented to address this problem. To use OCSP Must-Staple certificates must include an OCSP Must-Staple extension. Available OCSP responders, certificates that support OCSP Must-Staple, and browsers that enforce the OCSP Must-Staple extension are required to implement OCSP Must-Staple. Thus, a change for every player involved in certificate revocation is necessary. However, in an experiment conducted 36.8% of OCSP Responders had at least one outage that lasted for hours, only 0.02% of certificates support OCSP Must-Staple and the only browser to implement OCSP Must-Staple so far is Firefox. [3] Additionally, an attacker could perform a DoS attack targeted at the OCSP responders, making the website inaccessible during this period to clients. Furthermore, OCSP Must-Staple had problems with CA inconsistencies and bugs in server implementation. [10]

6.5. Comparison

To evaluate and compare the different approaches the following section takes a look at them through six cate-

	CRLite	Let's Revoke	CRT	OCSP Must-Staple
Efficiency	18 MB & 580 kB/day	2 MB & 114 kB/day	6.71 MB & 205 kB/day	1.3 kB/TLS handshake
Privacy	yes	yes	yes	yes
Auditability	yes	yes	yes	yes
Timeliness	1-2 days	1-2 days	1-2 days	4 days
Deployability	High	Medium	Medium	High
Failure Model	Hard-Fail	Hard-Fail	Hard-Fail	Soft-Fail

TABLE 1: Comparison Of The Different Methods [8], [10]

gories of measurement [8], [10]:
Efficiency is defined by bandwidth consumption for the client.
Timeliness Measures in which intervals the methods get updated.
Failure Model Evaluates how the client behaves when unable to get the revocation status of a certificate.
Privacy Does the approach ensure the privacy of the client?
Deployability How high are the deployment requirements?
Auditability Is the client able to audit the result for the revocation check?
The comparison of the different methods is in table 1.

7. Evaluation

In table 1 you can see the values for the different methods. When looking at efficiency, the first value is the initial download and the second value is the daily update to maintain the set. OCSP Must-Staple is the only method not having an initial download, but it needs 1.3 kB for every TLS handshake performed. This would impose a bandwidth burden on the client. Let's Revoke seems to be the best when it comes to efficiency, but CRT also has the option of using only revoked certificates which would be considerably lower with 1.92 kB & 0.21 kB/day. None of the methods exposes the privacy of the client, and all of them are auditable. There is also no difference in the timeliness of the methods with OCSP Must-Staple as the exception, which gets updated only every four days. When it comes to deployability, both CRLite and OCSP Must-Staple demand more change and effort being integrated into the existing infrastructure. Let's Revoke, and CTR does not demand much change and can be deployed with only demanding change from two actors in certificate revocation less. All of the methods hard-fail. OCSP Must-Staple also should hard-fail in theory, but that did not hold true when it was tested in real life. Having the categories of measurement in mind one might consider that Let's Revoke is the best choice. This is because of the size of the downloadable files and the deployability. However, when it comes to scalability and the ability to deal with mass revocation CTR is the more sensible choice.

8. Conclusion

This paper introduced certificate revocation and an overview of today's status quo for revocation validation and potential methods to improve the said standard. First, it was assessed how well the three different actors involved handle certificate revocation. The conclusion was that there is no established standard and revoked certificates get completely ignored by mobile browsers and there is no certainty for desktop browsers. Mozilla Firefox and Google Chrome both abandoned CRL and started deploying their CRLs: OneCRL and CRLSet. However, these only cover a tiny fraction of the present revoked certificates. Server administrators must manually revoke certificates which is not a fail-proof system and opens up time-frames of vulnerabilities. In the last years, approaches and methods of handling this problem were proposed. We looked at a few of them and assessed which one is best for real-life deployment. There are three different types of approaches: Pull-based, push-based, and network-assisted. The methods introduced were: CRLite (push-based), CRT (pull-based), Let's Revoke (push-based) and OCSP Must-Staple (network-assisted). Comparing these approaches led us to the conclusion that Let's Revoke is the most suitable method for real-life deployment at the moment. However, CRT is still in active development and might be more suitable in the future

References

[1] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. Maggs, A. Mislove, A. Schulman, G. Wilson, Christo Eason, B. Noble, and I. N. Sneddon, "An End-to-End Measurement of Certificate Revocation in the Web's PKI," *IMC '15: Proceedings of the 2015 Internet Measurement Conference*, pp. 183–196, 2015.

[2] K. Kiyawat, "Do Web Browsers Obey Best Practices When Validating Digital Certificates?" 2014.

[3] T. Chung, J. Lok, B. Chandrasekaran, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, J. Rula, N. Sullivan, and C. Wilson, "Is the web ready for ocsp must-staple?" *IMC '18: Proceedings of the Internet Measurement Conference 2018*, pp. 105–118, 2018.

[4] "What percentage of internet traffic is mobile?" https://www.oberlo.com/statistics/mobile-internet-traffic, accessed: 2021-03-22.

[5] "Usage share of operating systems," https://en.wikipedia.org/wiki/Usage_share_of_operating_systems, accessed: 2021-03-22.

[6] "Network security services," https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS, accessed: 2021-03-22.

[7] "Ca:revocationplan," https://wiki.allizom.org/CA:RevocationPlan, accessed: 2021-03-22.

[8] T. Smith, L. Dickinson, and K. Seamons, "Let's Revoke: Scalable Global Certificate Revocation," *Xu, Sadeghi (Hg.) 2020 – Proceedings 2020 Network and Distributed*, 2020.

[9] L. Zhang, D. Choffnes, D. Levin, T. Dumitras, A. Mislove, A. Schulman, and C. Wilson, "Analysis of SSL certificate reissues and revocations in the wake of heartbleed," *IMC '14: Proceedings of the 2014 Conference on Internet Measurement Conference*, pp. 489–502, 2014.

[10] L. Dickinson, T. Smith, and K. Seamons, "Leveraging locality of reference for certificate revocation," *ACSAC '19: Proceedings of the 35th Annual Computer Security Applications Conference*, pp. 514–528, 2019.

[11] J. Larisch, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson, "CRLite: A Scalable System for Pushing All TLS Revocations to All Browsers," *2017 IEEE Symposium on Security 52017*, pp. 539–5556, 2017.