Investigating how browsers implement certificate revocation checks in practice

Alexander Edberg
Linköping University
Linköping, Sweden
aleed476@student.liu.se

Kasper Landgren
Linköping University
Linköping, Sweden
kasla760@student.liu.se

***Abstract*—As the popularity of HTTPS rises, certificates become more and more common. It is important the certificates are handed out with care and if needed, revoked. Web browsers must keep track of revoked certificates in order to keep the user safe. Not only that, but the we browser should also not take a chance if it does not know the status of the served certificate. In this report, we investigate how Google Chrome and Firefox handle their revocation checks for different domains and different types of certificates.**

***Index Terms*—Certificate Revocation, CRL, OCSP**

I. Introduction

To ensure a safe browsing experience on the internet most of the traffic is encrypted using HTTPS (SSL/TLS). This encryption makes sure that no one can read or modify the communication between the client and server. Certificates are used to authenticate the server so that the client can be sure that they are talking to whom they want to communicate with. Certificate Authorities (CAs) are the ones giving out these certificates and are verifying that the server is who they claim to be. The certificates has an expire date when the owner of the certificate has to request a new certificate and once again verify who they are. However a certificate can also be revoked due to various reasons like a stolen private key.

When a certificate gets revoked it is important that the web browsers are aware of this since the certificate can no longer be trusted. If an attacker has access to a certificate they could easily eavesdrop on the communication until the certificate expires. Currently using a Certificate Revocation Lists (CRL) or Online Certificate Status Protocol (OCSP) are the most popular methods of keeping track of revoked certifications. It is important that the browsers do their revocation checks to make sure that the user is browsing safely. In this paper we look at how revocation checks are handled in practice by different browsers (Chrome and Firefox).

The goal of this paper is to answer the following questions:

- How do modern browsers perform revocation checks for different domains?
- Are different domains and certificates treated differently? If so, how?

The rest of this paper is structured as follows. Section II gives background on CAs, CR, CRL and OCSP. Section III describes our methodology for answering our questions. Section IV presents our result. Section V a discussion on our results and section VI will conclude the paper.

II. Background

A. Certificates

Certificates can be used in many ways, it can be used to sign and authenticate code, documents and individuals and organisations when browsing using HTTPS. The most used type of certificate when browsing with HTTPS is the x.509 certificate. This certificate validates that the individual or organisation is indeed who they claim to be [1]. There are different type of certificates that all require different types of validation before the certificate is handed out. Three of the most popular types of certificates are [2]:

- Domain Validation (DV) - A basic certificate that is given out if you can prove that you are the owner of the domain name.
- Organization Validation (OV) - This certificate require the same verification as DV but also includes validation of the person owning the domain name. Such as name and address of the person.
- Extended Validation (EV) - The hardest certificate to get, the validation for this certificate is quiet thorough and things like the legal, physical and operational existence of the organisation is verified.

CAs are the ones giving out these certificates and are usually a third party that both the owner of the certificate and the person putting their trust in the certificate trust.

A certificate that has been handed out by a CA can also be revoked by the same CA. A certificate can be revoked due to various reasons and down below we list the reason codes used in CRLs as described in RFC 5280 [3]:

- unspecified (0),
- keyCompromise (1),
- cACompromise (2),
- affiliationChanged (3),
- superseded (4),
- cessationOfOperation (5),
- certificateHold (6),
- – value 7 is not used,
- removeFromCRL (8),
- privilegeWithdrawn (9),

- aACompromise (10)

An expired certificate is not the same as a revoked one and should not exist inside a CRL. The expired certificate should be handled in the same way as a revoked one and not be trusted by the web browsers.

B. CRL

Certificate Revocation Lists (CRL) are one of the most popular methods of keeping track of revoked certificates. Its a way for CAs to keep track of and inform which certificates no longer can be trusted. The CRLs works as a type of blacklist and contains information about the revoked certificates such as serial number, date of revocation, reason for revocation and who issued the revocation. Once a certificate is added to the CRL it does not get removed from the list until its expired. CRLs are updated by the CAs and can be update as soon as a new certificate is added to a couple of days in between.

Although CRLs are very popular they have some disadvantages to them. There can be large amount of overhead since the client needs to download the CRL and then go through the whole list which can be a couple of thousand lines long to make sure that the certificate is not present in the list. If the client is unable to download the CRL in the first place then most clients will default to trust the certificate.

1) CRLSet: CRLSet is the CRL that chrome uses. This list is generated and used by chrome and revocations are added by crawling selected CRLs published by CAs. The CRLSet is intended to contain intermediate revocations and is updated infrequently (at most once every few hours) [4].

2) OneCRL: Since Firefox 37 was launched in 2015 Firefox has been using OneCRL which is Firefox own list of revoked certificates. OneCRL only covers intermediate certificates from the CAs in Mozilla's root program and is updated when a CA notifies about an update [5].

C. OCSP

Online Certificate Status Protocol (OCSP) is another popular way to keep track of revoked certificates. OCSP tries to make the overhead smaller for the client by not forcing the client to download the CRL and instead send a query to the CA asking for the status of the certificate. The CA returns a response of the certificate status (good, revoked, or unknown).

OCSP solves the overhead problem for the client but introduces a new possible overhead for the OCSP server since queries are sent for every single certificate. Every CA signs a lot of certificates and every time someone want to connect to a website with a certificate from the CA a request is made. OCSP can also be a privacy concern. Every time a query is sent the OCSP responder has access to the IP of who sent the query and what website they are visiting. This information could possibly be used to track users browsing behavior.

D. OCSP Stapling

OCSP stapling is a continuation of OCSP. In OCSP stapling the client does not have to make a request to a separate OCSP responder, instead the client receives a timestamped OCSP response from the web server directly. In this case the web server contacts the OCSP responer at certain intervals to validate its certificate [6]. This means that there is less strain on the client since it no longer has to make a separate request and receives the OCSP response in the handshake with the web server. This also improves privacy since the OCSP response is now served by the web server and not the OCSP responder.

E. Wireshark

Wireshark is a free and open-source software that allows it users to capture and analyse network traffic on its' computer. It has a lot of features and allows you to filter and dissect all the packets going through the clients network [7].

F. TLSv1.2

Transport Layer Security (TLS) is a security protocol that provides secure communications over a network. It is used in HTTPS to make sure that no one can read or modify the contents of the packets and also makes sure that the two parties communicating are who they claim to be by certificates [8]. TLS 1.2 is the most common version of TLS used today [9].

III. Methodology

We have been tracking IPv4 traffic over the protocol TLSv1.2 [10] using Wireshark [7]. All traffic has been logged and compared with revoked certificates in order to identify how often they are used.

We also read the Chromium source code [11] in order to try to find how exactly certificates are checked. We started by identifying the interesting parts of the source code and found the subfolder "cert" [12]. Through the README-file, we identified the interface CertVerifyProc which contained the interface used for verifying certificates. This interface contained a function called Verify(), which in turn uses many other function calls to verify certificates.

We tested the following websites on both Firefox and Google Chrome:

- https://revoked-rsa-dv.ssl.com/
- https://revoked-rsa-ev.ssl.com/
- https://revoked-ecc-dv.ssl.com/
- https://revoked-ecc-ev.ssl.com/

To investigate how/if the browsers use OCSP-checks we used wireshark to capture traffic and the four websites with revoked certificates to see if Google Chrome and Firefox sent any OCSP requests. We also tested certificates on badssl [13].

IV. Results

What we found is that normal daily browsing did not use any revoked certificates at all, and the only ones we caught were those we were knowingly trying to get past Google Chrome's CRLSet [14].

We had assumed that Chromium performs no OCSP-checks, but that appears to be incorrect, as we found the code in fig. 1 which contains a parameter named ocsp_response, which indicates that such a control is performed. We have not been able to establish how often these checks are performed.

```
int CertVerifyProc::Verify(X509Certificate* cert,
                           const std::string& hostname,
                           const std::string& ocsp_response,
                           const std::string& sct_list,
                           int flags,
                           CRLSet* crl_set,
                           const CertificateList& additional_trust_anchors,
                           CertVerifyResult* verify_result,
                           const NetLogWithSource& net_log) {
  net_log.BeginEvent(NetLogEventType::CERT_VERIFY_PROC, [&] {
    return CertVerifyParams(cert, hostname, ocsp_response, sct_list, flags,
                            crl_set, additional_trust_anchors);
  });
```

Fig. 1. The implementation of CertVerifyProc::Verify() in the Chromium source code

```
if (!ocsp_response.empty()) {
  dict.SetStringKey("ocsp_response",
                    PEMEncode(ocsp_response, "NETLOG OCSP RESPONSE"));
}
```

Fig. 2. Check for non-empty ocsp_response in CertVerifyParams()

TABLE I
BROWSER BEHAVIOR ON REVOKED WEBSITES.

Website	Chrome	Firefox OCSP enabled	Firefox OCSP disabled
revoked-rsa-dv.ssl.com	Not Revoked	Revoked	Not Revoked
revoked-rsa-ev.ssl.com	Revoked	Revoked	Not Revoked
revoked-ecc-dv.ssl.com	Not Revoked	Revoked	Not Revoked
revoked-ecc-ev.ssl.com	Revoked	Revoked	Not Revoked

We did some comparisons with Firefox and found that while Firefox caught every revoked certificate we tested, Google Chrome did not. From the tested websites listed above, only the revoked EV certificates, test number two and four, were caught by Google Chrome as revoked, while all four were caught by Firefox. However when disabling OCSP-checking in Firefox it did not catch any of the revoked certificates. We found that the pinning-test on badssl was caught by Google Chrome, and not by Firefox, but after updating to a newer version of Firefox, we found no difference between Firefox v88.0 and Google Chrome v90.0.4430.93 when testing certificates on badssl.

Fig. 2 shows that the function CertVerifyParams() accepts an empty OCSP response and will only add it to the parameter list if it exists.

From our findings Google Chrome never sent any OCSP requests. Firefox also never sent any when we disabled OCSP-checking in the settings. With OCSP-checking enabled in Firefox we found that Firefox do send a OCSP request upon its first visit to a new website. However a OCSP request was not sent the second time we visited the websites. After clearing the cache Firefox did still not send any OCSP requests when we visited the websites. Only once we restarted Firefox it started sending the OCSP requests again suggesting that Firefox might only send requests the first time its visiting a website in a new session.

V. DISCUSSION

Since we only checked Kasper's personal internet traffic, we can only extrapolate this data to persons with similar browsing habits. We also have a very small amount of data when compared to everybody who is using Google Chrome for their daily internet browsing.

We had difficulties pinpointing where exactly in the code these verifications are performed since it includes code from 38 other files from the Chromium library. Mapping many of these paths may have been possible if we had not started off with the intent of registering how often revoked certificates are used in practise, and instead started studying the source code from the beginning.

```
grep "Verify(" * | wc -l
grep: internal: Is a directory
296
```

Fig. 3. Occurrences of Verify in the cert directory

As shown in fig. 3, Verify appears 296 times in the cert directory alone. One of these is the declaration and some are comments.

```
grep -r "Verify(" . | wc -l
1491
```

Fig. 4. Occurrences of Verify in the Chromium source code

As shown in fig. 4, Verify appears 1491 times in the entire code base. Scanning the code base for references and occurrences is very time consuming. It takes approximately 45 minutes for every single scan of the code base to run to completion. This slowed our progress by a significant amount. This could be improved upon by narrowing down the search, which would require finding exactly which folders are of interest.

Given that ocsp_response is treated as an optional parameter, we can conclude that Chrome does not perform these checks at every call and that CRLSet is lacking revoked certificates since we have found revoked certificates that were allowed.

Since Google Chrome does not perform OCSP-checks by itself [15] and still caught revoked certificates, we can assume that these revoked certificates are all contained in CRLSet. There are, however, browser extensions to Google Chrome that add OCSP-checks to the browser. [16]

VI. CONCLUSION

How do modern browsers perform revocation checks for different domains? We have not found a definitive answer to this question.

Are different domains and certificates treated differently? If so, how? We have not found a definitive answer to this question.

ACKNOWLEDGMENT

We would like to thank the course staff of the course TDDD17 Information Security, Second Course at Linköping University, and especially our supervisor Niklas Carlsson, for their involvement and support during this project.

REFERENCES

[1] [Online]. Available: https://sectigo.com/resource-library/what-is-x509-certificate
[2] [Online]. Available: https://www.verisign.com/en_US/website-presence/online/ssl-certificates/index.xhtml
[3] [Online]. Available: https://datatracker.ietf.org/doc/html/rfc5280page-69
[4] [Online]. Available: https://dev.chromium.org/Home/chromium-security/crlsets
[5] [Online]. Available: https://blog.mozilla.org/security/2015/03/03/revoking-intermediate-certificates-introducing-onecrl/
[6] [Online]. Available: https://knowledge.digicert.com/quovadis/ssl-certificates/ssl-general-topics/what-is-ocsp-stapling.html
[7] [Online]. Available: https://www.wireshark.org/
[8] [Online]. Available: https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/
[9] [Online]. Available: https://www.calcomsoftware.com/leaving-tls1-2-using-tls1-3/
[10] [Online]. Available: https://www.ibm.com/docs/en/zvse/6.2?topic=openssl-transport-layer-security-tlsv12
[11] [Online]. Available: https://github.com/chromium/chromium
[12] [Online]. Available: https://github.com/chromium/chromium/blob/master/net/cert
[13] [Online]. Available: https://badssl.com/
[14] [Online]. Available: https://dev.chromium.org/Home/chromium-security/crlsets
[15] [Online]. Available: https://www.computerworld.com/article/2501274/google-chrome-will-no-longer-check-for-revoked-ssl-certificates-online.html
[16]