CRL Lifetime: Analysis of Credential Theft Mitigation and Risk to Operational Availability

Executive Summary

XXX

Table of Contents

1 Introduction 1

2 Terminology and Background Error! Bookmark not defined.

3 CRLs and “Expiration” 1

4 Analysis of CRL Lifetime Choice 3

5 A Possible “Soft Failure” Mode 3

6 Conclusions and Recommendations 3

7 References 4

Appendix A CRL Alternatives 5

Appendix B Some detailed text… 6

# Introduction and Background

## Certificate Revocation Lists (CRLs)

Certificate authorities (CAs) issues X.509 certificates (Section 4 of [1]) (in this document, simply “certificates”) that serve to authenticate (provide and prove a identity) entities to other entities. In this document, we will refer to entities using a certificate to authenticate as *end* entities and entities accepting certificates for authentication are know as *relying parties*.

CAs, during the course of operation, *revoke* certificates from time-to-time, that is, indicate to relying parties that a particular certificate is no longer valid and not to be trusted. Revocations can come about from normal management processes (e.g., since certificate are immutable once issued, a change in entity contact information typically results in a new certificate being issued and the old one being revoked), but can also occur in the event of a cybersecurity incident (e.g., a the cryptographic private key associated with a certificate is revealed to a party besides its legitimate owner), in which case revocation occurs to prevent misuse of the certificate.

The process of revocation can take several forms. This document focuses on certificate revocation lists (CRLs) (see Section 5 of [1]) with some alternatives described in 5Appendix A.

CRLs are created and published by CAs[[1]](#footnote-1) on a regular basis. Put simply, a CRL lists all the certificates that should be considered revoked. Relying parties regularly obtain the latest CRLs from CAs they trust to issue certificates. When a relying party validates a certificate, it verifies the certificate does not appear in the appropriate CRL; if a certificate does appears in the CRL, a relying party should not accept the certificate.

## CRLs Lifetime and Expiration

CRLs have a *nextUpdate* field that indicates the time by which the subsequent CRL should be published. The semantics of this field are not well defined, but this document will refer to a CRL whose nextUpdate time has past as being *expired*. The document will refer to the timespan from the issuance of a CRL until the nextUpdate time as the *CRL lifetime*.

When a relying party has, due to some failure in the CRL distribution process, an expired CRL, it faces a choice in how to behave: it can continuing business as normal using the expired CRL, accepting the risk there may be revoked certificates of which it is unaware, or it can enter a failure state, with all authentication failing, which often means ceasing operations for all practical purposes. Most software in the computational grid domain currently chooses the latter behavior, failing when an expired CRL is encountered.

The choice to continue operation would seem to reduce risk to availability, since a failure in the distribution of a CRL will not cause a problem for all the unrevoked certificates that should not be impacted.

The risk the failure mode addresses is as follows:

1. An attacker has obtained a certificate (and associated private key material) through some unauthorized means.
2. The fact that the attacker has obtained the certificate has been detected and the appropriate CA revokes the certificate by issuing a CRL containing the certificate.
3. However, the attacker, to keep using the certificate somehow blocks the CRL distribution process (either globally or just to a subset of relying parties, perhaps just one), so that relying parties will still be using an old CRL that doesn’t contain the revoked certificate.
4. The attacker can use the certificate until either the distribution of the CRL to relying parties is restored, or the certificate expires through its normal expiration mechanism.

By having CRLs expire and cause a failure mode, this mitigates the risk from a compromised certificate by bounding the amount of time an attacker can use a certificate when the CRL distribution fails[[2]](#footnote-2).

The trade-off for this mitigation is that any failure in CRL distribution due to non-malicious means causes an operational failure if not addressed before the CRL expires.

## Other CRL Attributes

Two other technical points regarding CRLs that are germane to this document:

* The absence of a CRL is not treated as a failure, but as an administrative decision not to do revocation (at least in the computational grid domain). Hence, the absence of a CRL does not cause the same failure mode as an expired CRL, so a typical remediation for failing to have a non-expired CRL is for an administrator to remove the expired CRL, essentially changing a service to the non-failure mode.
* CAs digitally CRLs when they create them. When a relying party checks a CRL, it verifies this signature. In practice this means a CRL cannot be created or modified by any other party and must be obtained from the CA.

# Analysis of CRL Lifetime Choice in Failure Mode

We turn now to an analysis of the choice of CRL lifetime (made by the CA), assuming relying parties are using CRLs and operating in a mode where they fail if only an expired CRL is available.

At high level, the CRL lifetime represents a trade-off in mitigating the threat of a stolen credential in combination with failed CRL distribution by bounding the time that credential can be used, versus a risk to operational availability in that errors in CRL distribution can cause services to effectively fail.

At one extreme, the CRL exceeds the lifetime of issued certificates, it adds no benefit, since a certificate acquired by an attack will expire before the CRL does. For standard certificates with lifetimes measured in years, it also seems to add little risk since the risk of CRL distribution being broken for so long seems unlikely.

At the other extreme, as CRL lifetime approaches zero, we have in effect OCSP (described in 5Appendix A) where the relying party makes a network query for every end entity it authenticates and any lack of availability of the CA causes a failure, but the window of misuse of certificate by an attacker is reduced to the minimum possible considering revocation.

Another data point is 1x106 seconds (approximately 11.5 days), which is the maximum lifetime of a short-lived certificate as defined by the International Grid Trust Federation [2]. This time is significant because much of the community using such certificates consider this maximum to be sufficient to mitigate risk such that revocation checking is not required.

# A Possible “Soft Failure” Mode

Currently software stacks “hard fail” when they encounter an expired CRL. An alternative would be a failure mode where an expired CRL is not a hard failure, but continued operation with strong logging as to alert an administrator who can take action as desired (e.g., based on other context, such as being in the middle of a science run). This would seem to be a desirable option at least for administrators to configure. At this time, the author knows of no software that allows for this administrative configuration.

# Conclusions and Recommendations

One recommendation is that software developers should offer a configuration option that allows a administrator to make an expired CRL a “soft fail” as described in Section 3, where the service continues to function but alerts the administrator via logging or other mechanisms. This would allow administrators to take off a lower risk to availability for an increased risk of misused credentials if the alert is not acted on in a timely fashion. If use of this option were found to be popular, it could become the default.

# References

1. D. Cooper, S. Santesson S. Farrell S. Boeyen, R. Housley, and W. Polk. RFC 5820: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. May 2008. https://www.ietf.org/rfc/rfc5280.txt
2. Profile for SLCS X.509 Public Key Certification Authorities with Secured Infrastructure Version 2.1b. February 3, 2009. <http://www.tagpma.org/files/SLCS-2.1b.pdf>
3. M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. RFC 2560: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP. June 1999. <https://www.ietf.org/rfc/rfc2560.txt>
4. Revocation checking and Chrome's CRL. February 5, 2012. http://www.imperialviolet.org/2012/02/05/crlsets.html

CRL Alternatives

We briefly describe some alternatives to CRLs and discuss how they relate to the analysis in this document.

* Online Certificate Status Protocol (OCSP) [3]: OCSP is, rather simply put, a query protocol that allows a relying party to ask a CA in real time whether a certificate is valid or not. At a high level, it is equivalent to CRLs as their lifetimes approach zero.
* Delta CRLs (also defined in Section 5 of [1]) allows for incremental CRLs that serve to reduce resource consumption when CRLs grow very large. For the analysis in this document, the use of delta CRLs is the same as normal CRLs.
* Use de-authorization instead: One option is to use an authorization mechanism instead of revocation to deny access to a user whose credential has been compromised. The challenge with this approach is that authorization typically works on a user’s identity, while revocation works on a specific certificate. This means that if a new certificate is issued to a user whose certificate was compromised, it wouldn’t work until the user is re-authorized, which entails being certain that the compromised certificate has been removed from the attackers control, which is difficult.
* Use software updates: Google and other browsers have moved to using software updated to push CRLs to end entities [4]. This presumably requires an efficient software update mechanism.

Some detailed text…

XXX Don’t know if I want this or not.

We consider a simple scenario with a single CA, relying party and end entity using a certificate to authenticate to that relying party. Following a typical process, the CA is publishing CRLs on a regular schedule[[3]](#footnote-3), and the relying party is regularly obtaining and using those CRLs. There are a set of time intervals in this process that interact to impact cybersecurity and operations:

* We define the interval between acts of the CA publishes a new CRL as tpublish.
* We define the time between the publication of a CRL and the time it expires (i.e., the time indicated by its nextUpdate field) as tlifetime.
* We define the interval between acts of the relying party obtaining the latest CRL as tdownload.

A few observations about requirements on the relationship of these times for the distribution system to function properly:

* Minimally, tpublish must be less than or equal to tlifetime, otherwise there will exist periods when only a expired CRL is published.
* Minimally, tdownload must be less than or equal to tlifetime, otherwise there will exist periods when the relying party has an expired certificate.

For resilience, tpublish and tdownload, should be much less than tlifetime such that tlifetime is several times the length of tpublish and tdownload. This provides for resilience since there needs to be not just one CRL publication or download failure, but multiple failures over tlifetime to cause an operational service failure.

1. CAs may delegate CRL creation to another entity, but for the purposes of this document, we consider that entity to be part of the CA and don’t distinguish the two. [↑](#footnote-ref-1)
2. An assumption is the lifetime of the CRL is less than the lifetime of the certificate. [↑](#footnote-ref-2)
3. As discussed subsequently, a CA may issue a CRL ad hoc in response to a cybersecurity incident that causes a certificate to be revoked. [↑](#footnote-ref-3)