Revocation protocols of WebPKI and Revocation Transparency

Nikita Korzhitskii
Niklas Carlsson
Lifecycle of a typical WebPKI certificate

1. Issuance
Certificate Authority -> Server

2. Use
Client <-> Server
Hello
Key exchange, communication
...  

3. Expiration
Certificate
Name: www.liu.se
From: January 1
To: February 30

Hello
Certificate
You shall not pass!

www.liu.se
Revoking a certificate

1. Revocation – is a process of invalidating a certificate prior its expiration.

   Let the private key of the certificate be compromised, and the certificate owner asks the CA to revoke the certificate. Then:

   Certificate Authority  
   Certificate + Revocation reason  
   Revocation status endpoint, R*  
   *OCSP and/or CRL

2. Status delivery
   February, 1\textsuperscript{st}

   Revocation status endpoint, R*  
   Certificate  
   OK?  
   Is  
   Revoked  
   Client  
   Hello  
   Server  
   You shall not pass!
**Certificate Revocation List (RFC 5280)**

http://ca.liu.se/revoked.crl

<table>
<thead>
<tr>
<th>serial#</th>
<th>[issuer]</th>
<th>[date]</th>
<th>[reasonCode]</th>
</tr>
</thead>
<tbody>
<tr>
<td>52</td>
<td>liu.se</td>
<td>Feb, 4</td>
<td>1</td>
</tr>
<tr>
<td>412</td>
<td>liu.se</td>
<td>Feb, 9</td>
<td>5</td>
</tr>
</tbody>
</table>

CRL date, next update, signature

*CRLs are being phased out.*
Online Certificate Status Protocol (RFC 6960)

Client → OCSP responder
- Serial#, H(issuerDN), H(issuerKey)
- Status, [Revocation Date], [Next update], [Signature]

http://ocsp.liu.se/

OCSP ”stapling” (RFC 6066, 6961)

Client → Server → OCSP responder
- Hello
- Certificate, Signed status
- Is Certificate OK?
- Good, Signed status

...
Steps in the process of checking revocation status with different protocols: (a) with CRLs, the client fetches the (potentially large) CRL after obtaining the certificate in the TLS handshake; (b) with OCSP, the client asks for the revocation status of only the particular certificate; (c) with OCSP Stapling, the server is supposed to prefetch the OCSP response and provide it in the handshake, and if it does not, the client can fetch the OCSP response as in (b); and (d) with OCSP Must-Staple, the server must provide an OCSP response in the handshake or the client will reject the certificate.
Revocation does not work

  - obtaining certificate status is expensive
  - most browsers don’t check certificate status
  - custom revocation set (CRLSet by Google) only covers 0.35% of all revocations
- Chung et al., *Is the Web Ready for OCSP Must-Staple?* IMC 2018
  - not yet
- Mass revocations happen, and their exact scale is unclear
  - Zhang et al., *Analysis of SSL Certificate Reissues and Revocations in the Wake of Heartbleed*, IMC 2014

Other issues with current revocation status protocols:
- Performance (OCSP, CRL)
- Availability (OCSP, CRL)
- Replay attacks (OCSP, CRL)
- Privacy (OCSP)
- Soft-fails (Browsers ignore failed status requests)
- Transparency
Revocation does not work – Fixes

Possible fixes:
- Must-staple
- Custom revocation sets (CRLSet, OneCRL, ...)
- Short validity periods
- A totally new WebPKI ...
  - Kubilay et al., *CertLedger: A new PKI model with Certificate Transparency based on blockchain*, 2019
- Revocation Transparency
Revocation Transparency

- Broadly, a mechanism for logging (and optionally, delivery) of revocations.
- Could be used to create up-to-date revocation sets, detect revocation-related misbehavior by CAs, immutably preserve revocation history.
- Several schemes, standalone or as a part of a new PKI:
  - *CertLedger* (Kubilay et al., 2019), *AKI* (Hyun-Jin Kim et al., 2013), *DTKI* (Yu et al., 2016), *CertChain* (Chen et al., 2018)

- Our research goal:
  - Motivate the need for Revocation Transparency through (an ongoing) measurement
  - Develop a feasible and low-deployment-cost Revocation Transparency scheme on top of existing Certificate Transparency
  - Compare with other proposals
Thank you!

www.liu.se