Online Certificate Status Protocol (OCSP)

University of Tartu

Spring 2018
Online Certificate Status Protocol

CRL shortcomings:

- Size of CRLs
- Client side complexity
- Outdated status information

“The Online Certificate Status Protocol (OCSP) enables applications to determine the (revocation) state of an identified certificate.”

- Where can the relying party find the OCSP responder?
- How is the certificate identified in the OCSP request?
- How is the integrity of OCSP response assured?
- How to ensure the freshness of OCSP response?
- At which point OCSP must be checked?
Authority Information Access

<table>
<thead>
<tr>
<th>Certificate Hierarchy</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ DigiCert High Assurance EV Root CA</td>
</tr>
<tr>
<td>▼ DigiCert SHA2 High Assurance Server CA</td>
</tr>
<tr>
<td>*.eesti.ee</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Certificate Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certificate Subject Key ID</td>
</tr>
<tr>
<td>Certificate Subject Alt Name</td>
</tr>
<tr>
<td>Certificate Key Usage</td>
</tr>
<tr>
<td>Extended Key Usage</td>
</tr>
<tr>
<td>CRL Distribution Points</td>
</tr>
<tr>
<td>Certificate Policies</td>
</tr>
<tr>
<td>Authority Information Access</td>
</tr>
<tr>
<td>Certificate Basic Constraints</td>
</tr>
<tr>
<td>Certificate Signature Algorithm</td>
</tr>
<tr>
<td>Certificate Signature Value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Critical</td>
</tr>
<tr>
<td>OCSP: URI: <a href="http://ocsp.digicert.com">http://ocsp.digicert.com</a></td>
</tr>
</tbody>
</table>
OCSP over HTTP

Stream Content

POST / HTTP/1.0
Content-Type: application/ocsp-request
Content-Length: 120

0v0t0M0K0I0...+....1...6.2\ch.-...a.I....4E=...00.>....
9.7w.+......#0I0+.+...0...)K...4".Z.T ]HTTP/1.0 200 Ok
last-modified: Wed, 07 Mar 2012 18:19:19 GMT
content-type: application/ocsp-response
content-transfer-encoding: binary
content-length: 1165
cache-control: max-age=514527, public, no-transform, must-revalidate
date: Thu, 08 Mar 2012 19:23:52 GMT
connection: close

0...
.0.~+....0...o0..k0......J0H1.0...U...US1.0...U.
..Thawte, Inc.1"0...U...Thawte SSL OCSP Responder...20120307181919Z0s0q0I0...+....1...6.2
\ch...a.I....4E=...00.>....
9.7w.+......20120307181919Z0...20120314181919Z0
*.H.

.R.)c>csh.4...t.j]WS....
ct...a.]"U.Y.V[.\...].D...%T...].o.(?
...@.aY...~...D...0\..U.j>....)U...415...-9....0...0...0...0...9....E...0
*.H.
...0<1.0...U...US1.0...U.
..Thawte, Inc.1.0...U...
Request Syntax

OCSPRequest ::= SEQUENCE {
  tbsRequest TBSRequest,
  optionalSignature [0] Signature OPTIONAL }

Signature ::= SEQUENCE {
  signatureAlgorithm AlgorithmIdentifier,
  signature BIT STRING,
  certs [0] SEQUENCE OF Certificate OPTIONAL }

TBSRequest ::= SEQUENCE {
  version [0] Version DEFAULT v1(0),
  requestorName [1] GeneralName OPTIONAL,
  requestList SEQUENCE OF SEQUENCE {
    reqCert CertID,
    singleRequestExtensions [0] Extensions OPTIONAL }
  requestExtensions [2] Extensions OPTIONAL }

CertID ::= SEQUENCE {
  hashAlgorithm AlgorithmIdentifier,
  issuerNameHash OCTET STRING, -- Hash of Issuer’s DN
  issuerKeyHash OCTET STRING, -- Hash of Issuer’s public key
  (i.e., hash of subjectPublicKey BIT STRING content)
  serialNumber CertificateSerialNumber }

Response Syntax

```
OCSPResponse ::= SEQUENCE {
    responseStatus          OCSPResponseStatus,
    responseBytes           [0] EXPLICIT ResponseBytes OPTIONAL }

OCSPResponseStatus ::= ENUMERATED {
    successful (0), --Response has valid confirmations
    malformedRequest (1), --Illegal confirmation request
    internalError (2), --Internal error in issuer
    tryLater (3), --Try again later
    (4) is not used
    sigRequired (5), --Must sign the request
    unauthorized (6) --Request unauthorized
}

ResponseBytes ::= SEQUENCE {
    responseType OBJECT IDENTIFIER, --id-pkix-ocsp-basic
    response OCTET STRING }
```

- responseBytes provided only if responseStatus is "successful"
- Note that responseStatus is not signed
Response Syntax

response ::= SEQUENCE {
    tbsResponseData ResponseData,
    signatureAlgorithm AlgorithmIdentifier,
    signature BIT STRING,
    certs [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL }

ResponseData ::= SEQUENCE {
    version [0] EXPLICIT Version DEFAULT v1,
    responderID [1] Name,
    producedAt GeneralizedTime,
    responses SEQUENCE OF SEQUENCE {
        certID CertID,
        certStatus CertStatus,
        thisUpdate GeneralizedTime,
        nextUpdate [0] EXPLICIT GeneralizedTime OPTIONAL,
        singleExtensions [1] EXPLICIT Extensions OPTIONAL }
    responseExtensions [1] EXPLICIT Extensions OPTIONAL }

CertStatus ::= CHOICE {
    good [0] IMPLICIT NULL,
    revoked [1] IMPLICIT SEQUENCE {
        revocationTime GeneralizedTime,
        revocationReason [0] EXPLICIT CRLReason OPTIONAL }
    unknown [2] IMPLICIT NULL }
Who signs OCSP response?

The key used to sign the response MUST belong to one of the following:

- CA who issued the certificate in question
- CA Authorized Responder who holds a specially marked certificate issued directly by the CA, indicating that the responder may issue OCSP responses for that CA
  - OCSP signing delegation SHALL be designated by the inclusion of `id-kp-OCSPSigning` flag in an `extendedKeyUsage` extension of the responder’s certificate
  - How to check the revocation status of this certificate?
- Trusted Responder whose public key is trusted by the requester
  - Trust must be established by some out-of-band means
How to provide response freshness?

- Include signed timestamp in OCSP response (producedAt and thisUpdate)
  - What should be the allowed time difference?
  - Replay attacks
  - System clock in the Trusted Computing Base
- Include nonce in the OCSP request and check it in the response
  - OCSP nonce extension (optional)
  - Prevents replay attacks
  - Vulnerable to downgrade attacks
Revocation checking in browsers

- CRLs are not supported
- Problems with OCSP:
  - Privacy leakage
  - Slower initial page loading
  - Chrome does not use OCSP – uses CRLSets
  - Firefox is not brave enough to fail-safe:

![Firefox about:config](image)

- Solution is OCSP stapling (web server provides OCSP response to the browser)
- How frequently the OCSP status should be queried?
import socket
sserv = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sserv.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
sserv.bind(('', 8888))
sserv.listen(0)

while True:
    (s, address) = sserv.accept()
    print "[+] Client %s:%s" % (address[0], address[1])

- **bind(('', 8888))** and **listen()** listens for client connections on all IPs on all network interfaces
- **accept()** will wait until client connects and returns tuple:
  - client socket (has **send()** and **recv()** methods)
  - address tuple – IP and port
- **SO_REUSEADDR** forces the kernel to reuse port even if it is in busy (TIME_WAIT) state (prevents error when rebinding)

http://docs.python.org/2/howto/sockets.html
Task: OCSP responder

Implement OCSP responder answering to HTTP POST requests.

$ ./ocspresponder.py
usage: ocspresponder.py --privkey privkey --cacert cacert --revoked cert [cert ...]

$ ./ocspresponder.py --privkey priv.pem --cacert rootCA.pem --revoked issued1.pem issued2.pem
  [+] Serial 1705406124 (issued1.pem) loaded
  [+] Serial 3532215973 (issued2.pem) loaded
  [+] Connection from 127.0.0.1:48318
  [+] Connection from 172.17.57.208:45394

$ openssl ocsp -url http://127.0.0.1:8888/ -no_nonce -VAfile rootCA.pem
  -issuer rootCA.pem -cert issued2.pem
Response verify OK
issued2.pem: revoked
  This Update: Apr 4 10:13:37 2018 GMT
  Reason: keyCompromise
  Revocation Time: Jan 1 12:12:00 2000 GMT

$ openssl ocsp -url http://127.0.0.1:8888/ -no_nonce -VAfile rootCA.pem
  -issuer rootCA.pem -cert issued3.pem
Response verify OK
issued3.pem: good
  This Update: Apr 4 10:13:39 2018 GMT

$ openssl ocsp -url http://127.0.0.1:8888/ -no_nonce -VAfile rootCA.pem
  -issuer rootCA2.pem -cert issued3.pem
Response verify OK
issued3.pem: unknown
  This Update: Apr 4 10:13:45 2018 GMT
Task: OCSP responder

- Bind your HTTP server on port 8888 on all network interfaces:
  
  $ netstat -na | grep 8888
  tcp 0 0 0.0.0.0:8888 0.0.0.0:* LISTEN

- HTTP server should be able to process sequential connections
- Must support only single certificate in OCSP request
- Signature check in load_serials() not required.
- Omit nextUpdate. Use current time for thisUpdate and producedAt.
- Return CertStatus:
  - “unknown” if not issued by CA (issuerNameHash or issuerKeyHash in CertID does not match).
  - “revoked” if certificate revoked (set arbitrary revocationTime and revocationReason).
  - “good” otherwise.
Task: OCSP responder


OCSP Request Data:
  Version: 1 (0x0)
  Requestor List:
    Certificate ID:
      Hash Algorithm: sha1
      Issuer Name Hash: 8350F92D60E6122B0112EF8E5381F3190BB2C703
      Issuer Key Hash: 4396CDBBB018CC4DF32D699971706FF1639B3BFB
      Serial Number: 577DC7A1

OCSP Response Data:
  OCSP Response Status: successful (0x0)
  Response Type: Basic OCSP Response
  Version: 1 (0x0)
  Responder Id: C = EE, O = University of Tartu, OU = IT dep, CN = Arnis Root CA
  Produced At: Apr 4 10:45:20 2018 GMT
  Responses:
    Certificate ID:
      Hash Algorithm: sha1
      Issuer Name Hash: 8350F92D60E6122B0112EF8E5381F3190BB2C703
      Issuer Key Hash: 4396CDBBB018CC4DF32D699971706FF1639B3BFB
      Serial Number: 577DC7A1
    Cert Status: good
    This Update: Apr 4 10:45:20 2018 GMT

  Signature Algorithm: sha1WithRSAEncryption
  Response verify OK
  issued3.pem: good
  This Update: Apr 4 10:45:20 2018 GMT
Task: OCSP responder

- **Bonus point for supporting nonce extension:**
  
  ```bash
  $ openssl ocsp -url http://127.0.0.1:8888/ -nonce -VAfile rootCA.pem
     -issuer rootCA.pem -cert issued3.pem
  Response verify OK
  issued3.pem: good
  This Update: Apr 4 10:29:35 2018 GMT
  ```

- **Half bonus point for returning response status “unauthorized” to clients from non-loopback IP (127.0.0.1):**
  
  ```bash
     -issuer rootCA.pem -cert issued3.pem
  Responder Error: unauthorized (6)
  ```

- **Half bonus point for serving nice response to GET requests:**
  
  ![Image of Firefox showing OCSP response]
  
  This server is processing only OCSP POST requests!
Hints

• Look on `resp_revoked.der`, `resp_good.der`, `resp_unknown.der`, `resp_nonce.der` and `resp_unauthorized.der` for response examples.
• Use `openssl -respout` parameter for debugging.
• Error “bad signature” may be caused by wrong DER encoding.
• DER encoding for ENUMERATED is the same as for INTEGER, just universal tag 10 instead of 2.
• DER encoding for GeneralizedTime is the same as for UTCTime, just 4-digit year encoding and universal tag 24 instead of 23.
• Datetime object conversion to GeneralizedTime string:
  ```python
datetime.datetime.utcnow().strftime("%Y%m%d%H%M%SZ")
```
• CertStatus is implicitly tagged NULL value (for good/unknown) or SEQUENCE (for revoked)
  • Implicit tagging replaces “type” byte of original value:
    • Class bits – context-defined (1 0)
    • Form bit – from value to be tagged (primitive/constructed)
    • Tag bits – tag number
Certificate Status

CRL and OCSP allows to verify if certificate valid now.

- Simple for authentication
- What about digital signature?
- Signature valid as long as certificate has not expired?

- Was the certificate valid at the time of signing?
- How to find out the time of signing?
Trusted Timestamping

Signed statement of timestamping authority (TSA):

> This data [data] was presented to me at this time: [time]
> Yours,
> --
> TSA
> [signature]


- data – usually a hash of the signature value
- Proves that the signature was given before the time specified
- Digital signature containers usually contain:
  - Signed files
  - Signature of files
  - Timestamp of the signature
  - OCSP response (acquired right after timestamping)
- How to verify digital signature after TSA/OCSP cert expires?
- Why is certificate suspension a bad idea?
DigiDoc Client

DigiDoc KLIENT

Container content:

Logistika Ja_Reiside_Assotsiatsiooni_leping.docx 21 KB

Signatures

Marika Priske
Signed on 09. September 2011 time 15:37
Signature is valid

Andres Valgeist
Signed on 09. September 2011 time 11:26
Signature is valid

Save files to disk

Send container to email
Browse container location

Print summary
Encrypt document

Add signature Close
XML Signature

<?xml version="1.0" encoding="UTF-8"?>
<SignedDoc format="DIGIDOC-XML" version="1.3" xmlns="http://www.sk.ee/DigiDoc/v1.3.0#">
  <DataFile Filename="document.doc" Id="D0">UEsDBQABgA...ASlEAAAA</DataFile>
  <Signature Id="S0">
    <SignedInfo>
      <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
      <Reference URI="#D0">
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>Q43ti5R/wgi8qOoHsygLFTXE0qU=</DigestValue>
      </Reference>
      <Reference URI="#S0-SignedProperties">
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>G0HmQqHCqMxULzfWSQfL2iOMIU</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue Id="S0-SIG">kgsCQ6...M4rkcj8=</SignatureValue> - signature of <SignedInfo>
    <X509Certificate>IID4z....V8APa</X509Certificate>
    <SignedProperties Id="S0-SignedProperties">
      <SigningCertificate>
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>RRFMpf0Xr5ZRYEs49m4S8M3oRnw</DigestValue>
      </SigningCertificate>
      <SignatureProductionPlace>
        <City>Tallinn</City>
      </SignatureProductionPlace>
    </SignedProperties>
    <OCSPValues>...
    </OCSPValues>
  </Signature>
</SignedDoc>
DigiDoc Crypto
XML Encryption

- Content encrypted using random 128-bit AES key
- Key encrypted using receivers RSA public key
- IV stored as a first ciphertext block
- Integrity protection not provided
Questions

• Where can the relying party find the OCSP responder?
• How is the certificate identified in the OCSP request?
• How is the integrity of OCSP response assured?
• How to ensure the freshness of OCSP response?
• At which point OCSP must be checked?
• What problem does the OCSP nonce extension solves?
• What is downgrade attack?
• What is needed in order to prove that the data was signed when the certificate was valid?