Online Certificate Status Protocol (OCSP)

University of Tartu

Spring 2016
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- Outdated status information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.”

- How to identify certificate?
- How to provide status information?
- How to provide response authenticity?
- How to provide response freshness?
- Where to obtain address of OCSP responder?
- At which point OCSP must be checked?
Authority Information Access

Certificate Hierarchy

- DigiCert High Assurance EV Root CA
- DigiCert SHA2 High Assurance Server CA
- *.eesti.ee

Certificate Fields

- Certificate Subject Key ID
- Certificate Subject Alt Name
- Certificate Key Usage
- Extended Key Usage
- CRL Distribution Points
- Certificate Policies
- Authority Information Access
- Certificate Basic Constraints
- Certificate Signature Algorithm
- Certificate Signature Value

Field Value

- Not Critical
- OCSP: URI: http://ocsp.digicert.com
- CA Issuers: URI: http://cacerts.digicert.com
  /DigiCertSHA2HighAssuranceServerCA.crt
OCSP over HTTP

Stream Content:

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

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

...
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 }
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
- Trusted Responder whose public key is trusted by the requester
  - Trust must be established by out-of-band means
- CA Designated Responder (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 in an extendedKeyUsage certificate extension included in the OCSP response signer’s certificate. This certificate MUST be issued directly by the CA that issued the certificate in question.
How to provide response freshness?

- Include signed timestamp in OCSP response (\textit{producedAt} and \textit{thisUpdate})
  - What should be the allowed time difference?
  - System clock in the Trusted Computing Base
  - Replay attacks
- Include nonce in the OCSP request and check it in the response
  - OCSP nonce extension (optional)
  - Prevents replay attacks
  - Downgrade attacks
Revocation Checking in Browsers

- CRLs used only when configured manually:
  - Problems with OCSP:
    - Slower initial page loading
    - Privacy leakage
    - Browsers are not brave enough to fail-safe:

- Solution – OCSP stapling
**Lightweight Directory Access Protocol (LDAP)**

“Directory services may provide any organized set of records, often with a hierarchical structure, such as a corporate email directory. Similarly, a telephone directory is a list of subscribers with an address and a phone number.”


- Used in PKI by CA to store CRLs and issued certificates

```
$ ldapsearch -x -h ldap.sk.ee -b c=EE "(serialNumber=38608050013)"
# LDAPv3
# base <c=EE> with scope subtree
# filter: (serialNumber=38608050013)
# requesting: ALL

# PAR\C5\A00VS\2CARNIS\2C38608050013, authentication, ESTEID, EE
dn:: Y249UEFSxaBPV1NcMkNBuk5JU1wyQz4MnjA4MDUwMDEzLG91PWF1dGlhbnRpY2F0awW9uLG99R
   VNURU1ELGM9RUU=
cn:: UEFSxaBPV1MsQVJ0SVMsMzg2MDgwNTAwMTM=
serialNumber: 38608050013
userCertificate;binary:: MIIEGDCCAwCgAwIBAgIETH9q6zANBgkqhkiG9w0BAQUFADBbMQswC
   UFssRJJ0IN3SAqG0kpoN HKwa62hC/ikwl/N5elkLVXLFJ9yZES3Ab2zy1gyPpfjAy1i/xJ1Gs
   Qx9nbFl02n75ZKx7+drgQ7OLfkhTaZDn6diKbr7WkzDYtE8bnK46jzvR55ZfVeBcFlGR0akj3
   UFssRJJ0IN3SAqG0kpoNDkwa62hC/ikwl/N5elkLVXLFJ9yZES3Ab2zy1gyPpfjAy1i/xJ1Gs
   rXBJXrayiUtNbcJwYRxR4xTtpfR0+qisQkDYzTwuVzxc800
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson

# PAR\C5\A00VS\2CARNIS\2C38608050013, digital signature, ESTEID, EE
dn:: Y249UEFSxaBPV1NcMkNBuk5JU1wyQz4MnjA4MDUwMDEzLG91PWRpZ210YWwgc2lnbmF0dXJLL
   G99RVNURU1ELGM9RUU=
cn:: UEFSxaBPV1MsQVJ0SVMsMzg2MDgwNTAwMTM=
serialNumber: 38608050013
userCertificate;binary:: MIIEGDCCAwCgAwIBAgIETH9q6zANBgkqhkiG9w0BAQUFADBbMQswC
   UFssRJJ0IN3SAqG0kpoN HKwa62hC/ikwl/N5elkLVXLFJ9yZES3Ab2zy1gyPpfjAy1i/xJ1Gs
   Qx9nbFl02n75ZKx7+drgQ7OLfkhTaZDn6diKbr7WkzDYtE8bnK46jzvR55ZfVeBcFlGR0akj3
   UFssRJJ0IN3SAqG0kpoNDkwa62hC/ikwl/N5elkLVXLFJ9yZES3Ab2zy1gyPpfjAy1i/xJ1Gs
   rXBJXrayiUtNbcJwYRxR4xTtpfR0+qisQkDYzTwuVzxc800
```

Lightweight Directory Access Protocol (LDAP)

“The certificates issued under this CP shall be published in public directory at ldap://ldap.sk.ee subsequent to the activation of the certificates. Suspended and revoked certificates are deleted from the public directory. In case of termination of suspension of certificates, the certificates shall be republished in the public directory. Expired certificates shall be deleted from the public directory on the date subsequent to the date of certificate expiry.” (from SK CP)

- Using LDAP for revocation checks:
  - Pros: can query for full certificate not just a serial number
  - Cons: LDAP traffic is not cryptographically protected
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

  Response verify OK
  issued2.pem: revoked
    This Update: Mar 31 22:13:37 2016 GMT
    Reason: keyCompromise
    Revocation Time: Jan 1 12:12:00 2000 GMT

  Response verify OK
  issued3.pem: good
    This Update: Mar 31 22:13:39 2016 GMT

  Response verify OK
  issued3.pem: unknown
    This Update: Mar 31 22:13:45 2016 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.

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: Mar 31 22:45:20 2016 GMT
Responses:
Certificate ID:
  Hash Algorithm: sha1
  Issuer Name Hash: 8350F92D60E6122B0112EF8E5381F3190BB2C703
  Issuer Key Hash: 4396CDBBB018CC4DF32D699971706FF1639B3BFB
  Serial Number: 577DC7A1
Cert Status: good
This Update: Mar 31 22:45:20 2016 GMT

Signature Algorithm: sha1WithRSAEncryption
Response verify OK
issued3.pem: good
This Update: Mar 31 22:45:20 2016 GMT
Task: OCSP responder

- **Bonus point for supporting nonce extension:**
  ```
  $ 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: Mar 31 22:29:35 2016 GMT
  ```

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

- **Half bonus point for serving nice response to GET 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: is the certificate valid now?

- Enough for digital 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?

![Graph showing the timeline of certificate status]

- Signer certificate valid
- CA certificate valid
- Message signed
- Signature valid
- Signer PK certified
- CA certificate created
Trusted Timestamping

Signed statement of timestamping authority (TSA):

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


- data – usually a hash of the signature value
- Proves that the signature was given at the time \( \leq \) time
- Digital signature containers usually contain:
  - Digital document
  - Signature of the document
  - Timestamp of the signature
  - OCSP response (acquired at the time of timestamping)
- How to verify digital signature after TSA cert expires?
- Why is certificate suspension a bad idea?
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/wgi8qOoHsygLFTXEqU=</DigestValue>
      </Reference>
      <Reference URI="#S0-SignedProperties">
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>G0HmQqHCqMxULzfWSOIL2i0mIU=</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue Id="S0-SIG">kgsCQ6...M4rkcj8=</SignatureValue> - signature of <SignedInfo>
  <X509Certificate>ID4z....V8APa</X509Certificate>
  <SignedProperties Id="S0-SignedProperties">
  </SignedProperties>
  <SigningCertificate>
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <DigestValue>RRFMpf0Xr5ZryEs49m4S8M3oRnw=</DigestValue>
  </SigningCertificate>
  <SignatureProductionPlace>
    <City>Tallinn</City>
  </SignatureProductionPlace>
</Signature>
  <OCSPValues>
    ...
  </OCSPValues>
</SignedDoc>
DigiDoc Crypto

[Image of the DigiDoc Crypto software interface with the following details:
Name: Arnis Paršovs
Personal code: 38608050013
Card number: E0044843
Auth certificate is valid

Container content:
- hello.txt

Keys:
- PARŠOVS, ARNIS, 38608050013, ID-CARD

Options:
- Send container to email
- Browse container location
- Decrypt
- Close]
XML Encryption

<?xml version="1.0" encoding="UTF-8" ?>
<enc:EncryptedData xmlns:enc="http://www.w3.org/2001/04/xmlenc#">
    <enc:EncryptedKey Recipient="PARŠOVS, ARNIS, 38608050013, ID-CARD">
        <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
        <ds:X509Certificate>
            A1UdEQQaM...sQkDYZTwuVzxc8O0</ds:X509Certificate>
        <enc:CipherValue>AZrh0j...Q0Y=</enc:CipherValue>
    </enc:EncryptedKey>

    <enc:EncryptedKey Recipient="...">
        <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
        <enc:CipherValue>QGKSSP...Wg20fGDSJal/Dr6Y1x4Fgg==</enc:CipherValue>
    </enc:EncryptedKey>

    <enc:EncryptionProperties>
        <enc:EncryptionProperty Name="LibraryVersion">CDigiDoc|3.8.0.1136</enc:EncryptionProperty>
        <enc:EncryptionProperty Name="DocumentFormat">ENCDOC-XML|1.0</enc:EncryptionProperty>
        <enc:EncryptionProperty Name="orig_file">hello.txt|21||D0</enc:EncryptionProperty>
    </enc:EncryptionProperties>
</enc:EncryptedData>

- 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