Network Working Group
Internet-Draft
Category: Best Current Practice
Intended status: Informational
Expires: May 25, 2014

W. Hoehlhubmer
Nov 25, 2013

Informational Add-on for HTTP over the Secure Sockets Layer (SSL) Protocol and/or the Transport Layer Security (TLS) Protocol draft-hoehlhubmer-https-addon-07

Abstract

This document describes an Add-on for websites providing encrypted connectivity (HTTP over TLS).

The Add-on has two parts, one for the Domain Name System (DNS) - storing the X.509 certificate hashes - and one for the webserver itself - an additional webpage providing specific informations.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 25, 2014.

Copyright Notice

Table of Contents

1. Introduction

HTTP over TLS [HTTPTLS] is not limited to e.g. electronic banking sites. E-commerce is also using this technology on their websites for encrypted communication between clients (users) and them.

A list of a few encryption algorithms:

(1) Advanced Encryption Standard (AES)
(2) Data Encryption Standard (DES, 3DES)
(3) Ron’s Code 4 (RC4)
(4) ...

As an example a list of some kinds of the Camellia encryption algorithm [CAMELLIA] (names taken from OpenSSL help [OPENSSL]):

(1) CAMELLIA-128-CBC: 128-bit Camellia encryption in CBC mode
(2) CAMELLIA-128-ECB: 128-bit Camellia encryption in ECB mode
(3) CAMELLIA-192-CBC: 192-bit Camellia encryption in CBC mode
(4) CAMELLIA-192-ECB: 192-bit Camellia encryption in ECB mode
(5) CAMELLIA-256-CBC: 256-bit Camellia encryption in CBC mode
(6) CAMELLIA-256-ECB: 256-bit Camellia encryption in ECB mode
(7) ...

A list of possible secure layer used:

(1) The Secure Sockets Layer (SSL) Protocol:
    (1a) Version 2.0 [SSLv2]
    (1b) Version 3.0 [SSLv3]

(2) The Transport Layer Security (TLS) Protocol:
    (2a) Version 1.0 [TLSv1.0]
    (2b) Version 1.1 [TLSv1.1]
    (2c) Version 1.2 [TLSv1.2]

A list of possible Ciphersuites for Transport Layer Security (TLS):

(1) Pre-Shared Key Cipher Suites [RFC4279]
(2) Elliptic Curve Cryptography (ECC) Cipher Suites [RFC4492]
(3) Pre-Shared Key Cipher Suites with NULL Encryption [RFC4785]
(4) AES Galois Counter Mode (GCM) Cipher Suites [RFC5288]
(5) DES and IDEA Cipher Suites [RFC5469]
(6) ECDHE_PSK Cipher Suites [RFC5489]
(7) Camellia Cipher Suites [RFC5932]
(8) ...

A list of possible Hashing Algorithms:

(1) the [MD2] Message-Digest Algorithm (historic see [RFC6149])
(2) the [MD4] Message-Digest Algorithm (historic see [RFC6150])
(3) the [MD5] Message-Digest Algorithm used commonly in past
(4) the US Secure Hash Algorithm 1 [SHA1]
(5) more US Secure Hash Algorithms [RFC6234]
(6) ...

Only the X.509 Certificates [RFC5280] are static, all other informations depend on the capabilities of the used web browser.

Not every browser allows you to view all these informations, especially the Cipher Suite the browser has picked for use.

With most browsers you can view the used X.509 certificates of the actual session, but you have no direct comparison if they are the correct ones.

It is a good practice to show these informations on the website.

The X.509 certificates which are shown by the browser and those, that are shown in this Add-on webpage MUST match; with other words:

If they do not match, there is going on a man-in-the-middle attack.

To give the browser, a plug-in, or just a stand-alone program, the ability to verify, that the X.509 certificate is correct, the Fingerprint and/or Hash of the X.509 certificates and also some additional informations for the community itself are stored in the Domain Name System (DNS) [DNS-1,DNS-2] as arbitrary string attributes as specified in [RFC1464].

This SHOULD be seen as an additional specification of the DNS-Based Authentication of Named Entities (DANE) [RFC6698] that increases the confidence and gives extended informations that can easily be read by the community.

1.1. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Implementing this Add-on

This Add-on has two parts.

Section 2.1. describes the implementation of the neccessary entries in the Domain Name System (DNS).

Section 2.2. describes the implementation of the additional webpage.

2.1. Implementing the DNS part

This part is implemented as an arbitrary string attribute, that has at least the following content:

(1) The Hashes of all X.509 certificates of the whole certificate chain
(2) The number of all X.509 certificates of the whole certificate chain
(3) The used Hashing algorithm as specified in [SHA1] or [RFC6234]:
    (3a) "sha1" for SHA-1
    (3b) "sha224" for SHA-224
    (3c) "sha256" for SHA-256
    (3d) "sha384" for SHA-384
    (3e) "sha512" for SHA-512
(4) The way the Hashing values were calculated:
    (4a) "0" for Non-packed Base64 encoded Hashes, see Section 2.1.1.
    (4b) "1" for Packed Base64 encoded Hashes, see Section 2.1.1.1.
(5) The Date and Time the X.509 certificate is valid, for format see Section 2.1.2.

For Syntax see Section 2.1.3.

2.1.1 Calculating the Hashes

For calculating the hashes use either [SHA1] or SHA-224, SHA-256, SHA-384, or SHA-512 as specified in [RFC6234].

Take each X.509 certificate of the whole chain and calculate the hash of the DER-encoded certificate.

The example certificates of Appendix A give these SHA-224 hashes in hex:

CA: 00fcc1bb4d09a392f5729a0c1e1ed4247db6b21da1fca9bf6d218db4
SSL: eacbdc6c27cba4ecc87b4e953b56c6987d87430b682b1f13031b04de

and these SHA-512 hashes in hex:

```
CA:  6744023893a9a046e713b5615bcf1a267a41da13712f4eb964e496754bd9
     43105a5a3a8b9b071dea25f90fa7aa9c877dcc2ec81a7c97b640b30dd350
     83252078
SSL:  df0dee228b19aa1eac6d2227d11cb243562058db5a4041b208ed7702869
      98747ed7ba08026791961d338cb2063f3485ec9fe07d5631a8a1b1da340
      25cb8962f5
```

Concatenate the binary form of the calculated hashes in the correct order beginning at the root.
Generate the Base64 encoding [RFC4648] from the concatenated hashes.

This example gives the following Base64 from the concatenated SHA-224 hashes:

```
APzBu00Jo5L1cpoMHh7UJH22sh2h/Km/bSGNtOrL3Gwny6TsyHtOlTtWxph9h0ML
aCsfEwMbBN4=
```

and from the SHA-512 hashes:

```
Z0QCOJOpoEbnE7VhW88aJnpB2hNxL065ZOSWdUvZQxBaWjqLmwcd6iX5D6eqnId9
zC7IGnyXtkCzDdNQgyUgeN8N7iKLGaoerG0iJ9EcskNWIFjbWkBBsgjtdwKGmYdH
7XuggCZ5GWHTOMsgY/NIXsn+B9VjGoobHaNAJcuJYvU=
```

Due to size limitations as specified in [DNS-2] Section 2.3.4. and the Syntax as specified in Section 2.1.2. below
this Base64 encoded hash MUST NOT be longer than 196 octets.

This table shows when to use the packed form of calculation explained in next Section 2.1.1.1.

Hashing algorithm	X.509 certificates
SHA-1	8 or more
SHA-224	6 or more
SHA-256	5 or more
SHA-384	4 or more
SHA-512	3 or more

Using of the non packed form SHOULD be preferred.

2.1.1.1. Calculating the Packed form

The calculation is the same except, that the binary form of the concatenated hashes is hashed again using the SHA-512 algorithm.

Generate the Base64 encoding from this SHA-512 hash.

The example from the previous Section 2.1.1. has only two X.509 certificates. There would not be any need of packing this by hashing again.

The Base64 encoding of this packed SHA-512 hash is the following:

```
4iBTHcxpK4GG0thWbLaq9gQx2UmFDPI2DJDWyeKYk3RmUwS+nkuCXYXR6ED4iGy4
Ftl5nFcsta9rwMvsaQx/wg==
```

> NOTE: The algorithm attribute refers to the calculation method of the X.509 certificate hashes. For calculation of the packed form there is always used the SHA-512 hashing algorithm.

2.1.2. Formatting the Date and Time

The date and time is formatted in GeneralizedTime as specified in [RFC5280] Section 4.1.2.5.2.

For the purposes here, GeneralizedTime values MUST be expressed in Greenwich Mean Time (Zulu) and MUST include seconds (i.e., times are YYYYMMDDHHMMSSZ), even where the number of seconds is zero. GeneralizedTime values MUST NOT include fractional seconds.

2.1.3. Arbitrary String Attribute Syntax

The syntax for a complete arbitrary string attribute, using the ABNF notation and core rules of [RFC5234], is:

```
attribute = DQUOTE
            attr-algo 1*SP     ; (3)
            attr-count 1*SP    ; (2)
            attr-packed 1*SP   ; (4)
            attr-time 1*SP     ; (5)
            attr-hashes 1*SP   ; (1)
            DQUOTE

attr-algo = "a=" hash-algo ";"
attr-count = "c=" cert-count ";"
attr-packed = "f=" packed-form ";"
attr-time = "v=" valid-time-from "-" valid-time-to ";"
attr-hashes = "x=" cert-hashes ";"
```

```
      cert-count = DIGIT  ; number of X.509 certificates of
                          ;  the whole certificates chain

      cert-hashes = base64-string
                          ; base64 encoding of the certificates
                          ;  hashes

      hash-algo = 1*("sha1" / "sha224" / "sha256" / "sha384" /
                     "sha512")

      packed-form = 1BIT  ; "0" non-packed base64 encoded hashes,
                          ; "1" packed base64 encoded hashes

      valid-time-from = time-str
      valid-time-to = time-str

      base64-string = 1*(ALPHA / DIGIT / "+" / "/") [ "=" [ "=" ] ]

      time-str = time-year time-month time-day time-hours time-minutes
                 time-seconds "Z"

      time-year = 4DIGIT
      time-month = 2DIGIT
      time-day = 2DIGIT
      time-hours = 2DIGIT
      time-minutes = 2DIGIT
      time-seconds = 2DIGIT
```

The example from Section 2.1.1. gives these:

```
"a=SHA224; c=2; f=0; v=19700101000000Z-19701231235959Z; x=APzBu0
0Jo5L1cpoMHh7UJH22sh2h/Km/bSGNtOrL3Gwny6TsyHtOlTtWxph9h0MLaCsfEw
MbBN4=;"

"a=SHA512; c=2; f=0; v=19700101000000Z-19701231235959Z; x=Z0QCOJ
OpoEbnE7VhW88aJnpB2hNxL065ZOSWdUvZQxBaWjqLmwcd6iX5D6eqnId9zC7IGn
yXtkCzDdNQgyUgeN8N7iKLGaoerG0iJ9EcskNWIFjbWkBBsgjtdwKGmYdH7XuggC
Z5GWHTOMsgY/NIXsn+B9VjGoobHaNAJcuJYvU=;"

"a=SHA512; c=2; f=1; v=19700101000000Z-19701231235959Z; x=4iBTHc
xpK4GG0thWbLaq9gQx2UmFDPI2DJDWyeKYk3RmUwS+nkuCXYXR6ED4iGy4Ftl5nF
csta9rwMvsaQx/wg==;"
```

All three are valid.

2.1.4. DNS-entry Namespace

For this Add-on a subdomain named "_sslinfo" is used.

> INFORMATIVE OPERATIONAL NOTE: Wildcard DNS records (e.g., *._sslinfo.example.com) are only used in context with Wildcard X.509 certificates. Note also that wildcards within domains (e.g., s._sslinfo.*.example.com) are not supported by the DNS.

The DNS entries in the Zone file for this example look like these:

```
; IPv4 address
www.example.com.           IN A 192.0.2.1

; IPv6 address
www.example.com.           IN AAAA 2001:db8::1

; X.509 certificates hashes, SHA-224
www._sslinfo.example.com.  IN TXT "a=SHA224; c=2; f=0; v=1970010
1000000Z-19701231235959Z; x=APzBu00Jo5L1cpoMHh7UJH22sh2h/Km/bSGN
tOrL3Gwny6TsyHtOlTtWxph9h0MLaCsfEwMbBN4=;"

; X.509 certificates hashes, SHA-512
www._sslinfo.example.com.  IN TXT "a=SHA512; c=2; f=0; v=1970010
1000000Z-19701231235959Z; x=Z0QCOJOpoEbnE7VhW88aJnpB2hNxL065ZOSW
dUvZQxBaWjqLmwcd6iX5D6eqnId9zC7IGnyXtkCzDdNQgyUgeN8N7iKLGaoerG0i
J9EcskNWIFjbWkBBsgjtdwKGmYdH7XuggCZ5GWHTOMsgY/NIXsn+B9VjGoobHaNA
JcuJYvU=;"

; X.509 certificates hashes, SHA-512, packed
www._sslinfo.example.com.  IN TXT "a=SHA512; c=2; f=1; v=1970010
1000000Z-19701231235959Z; x=4iBTHcxpK4GG0thWbLaq9gQx2UmFDPI2DJDW
yeKYk3RmUwS+nkuCXYXR6ED4iGy4Ftl5nFcsta9rwMvsaQx/wg==;"
```

2.2. Implementing the HTTP part

This Add-on is just one page of the website. Its content MUST be completely generated on server side. The Common Gateway Interface [CGI1.1] is RECOMMENDED to be used. There MUST exist at least one relative reference to this page as defined in [RFC3986] Section 4.2.

See Section 2.2.1. for the neccessary content of this webpage.

For doing so see the sample scripts from Appendix B.
To see how this Add-on works, see [MYADDON].

2.2.1. Webpage Content

The informations MUST be the following:

(1) The actual date and time

(2) The cipher specification name

(3) Number of cipher bits (actually used)
(4) Number of cipher bits (possible)

(5) The SSL Protocol version: SSLv2, SSLv3,
TLSv1.0, TLSv1.1, TLSv1.2, ...

(6) If cipher is an export cipher: false, true
(7) If secure renegotiation is supported: false, true

(8) Algorithm used for the public key of server's certificate
(9) Algorithm used for the signature of server's certificate
(10) Issuer DN of server's certificate
(11) Subject DN in server's certificate
(12) The serial of the server certificate
(13) The version of the server certificate
(14) Validity of server's certificate (start time)
(15) Validity of server's certificate (end time)

(16) Client certificate verification:
NONE, SUCCESS, GENEROUS or FAILED:reason

(17) SSL compression method negotiated: NULL when disabled

For connections where X.509 certificates are used for authentication these informations are RECOMMENDED:

(18) Algorithm used for the public key of client's certificate
(19) Algorithm used for the signature of client's certificate
(20) Issuer DN of client's certificate
(21) Subject DN in client's certificate
(22) The serial of the client certificate
(23) The version of the client certificate
(24) Validity of client's certificate (start time)
(25) Validity of client's certificate (end time)
(26) Number of days until client's certificate expires

This information MAY be given:

(27) The hex-encoded SSL session id
(28) Contents of the SNI TLS extension (if supplied with ClientHello)

These OPTIONAL informations depend on the used software:

(29) The SSL-module program version: e.g. Apache mod_ssl version
(30) The SSL program version: e.g. OpenSSL version

See Appendix C for a sample content.

2.2.2. Formating and Presenting the Webpage

You SHALL present this information simple, plain Text is enough. When using HTML, only relative references as defined in [RFC3986] Section 4.2. MAY be used. It is RECOMMENDED to use only a subset of [HTML2.0].

The actual date and time SHALL be formatted as specified in [RFC5322] Section 3.3. The time MUST NOT differ more than 5 seconds from the real date/time.

Any translation or sorting the order of this content is OPTIONAL.

Consider using either one of the charactersets as specified in [ISO8859] or the UTF-8 characterset as specified in [ISO10646].

3. DNS part Details

3.1. Handling Certificate Rollover

This is analogous to [RFC6698] Section A.4.

Suppose www.example.com has a single TXT record:

```
www._sslinfo.example.com.  IN TXT "a=SHA224; ...; x=Z0QCO..."
```

To start the rollover process, obtain or generate the new certificate or SubjectPublicKeyInfo to be used after the rollover and generate the new TXT record. Add that record alongside the old one:

```
www._sslinfo.example.com.  IN TXT "a=SHA224; ...; x=Z0QCO..."
www._sslinfo.example.com.  IN TXT "a=SHA384; ...; x=A078x..."
```

After the new records have propagated to the authoritative nameservers and the TTL of the old record has expired, switch to the new certificate on the Web server. Once this has occurred, the old TXT record can be removed:

```
www._sslinfo.example.com.  IN TXT "a=SHA384; ...; x=A078x..."
```

This completes the certificate rollover.

3.2. Verification Procedure

When the webbrowser or a plug-in honors the additional DNS entries, it SHOULD give a warning to the user:

(1) when it doesn’t find the entry
(2) when the entry doesn’t match

In case the DNS entries were retrieved by [DNSSEC] instead of simple DNS, then the user MUST give a permission to go further, when one of the two scenarios occurs.

The user MAY be warned, but MUST NOT be prevented to use the website.

4. IANA Considerations

There are no requests for IANA actions in this document.

5. Security Considerations

When implementing the HTTP part as a popup window in the browser, this information MUST also be available with enabled popup-blocker.

The implementation MUST NOT use any scripts, that run on client side: e.g. Javascript, ...

There SHOULD also be no references to other websites inside this Add-on page.

6. Acknowledgements

7. Recommendations

[DNSSEC] SHOULD be used for the DNS part.

Using a standardized URL for the HTTP part is RECOMMENDED, for more see Discussions at Section 9.

8. References

8.1. Normative References

[DNS-1] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[DNS-2] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

8.2. Informative References

[CAMELLIA] Matsui, M., Nakajima, J., and S. Moriai, "A Description of the Camellia Encryption Algorithm", RFC 3713, April 2004.

[DNSSEC] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[HTTPTLS] Rescorla, E., "HTTP over TLS", RFC 2818, May 2000.

[ISO8859] ISO/IEC 8859:1998. Information technology -- 8-bit single-byte coded graphic character sets.

[ISO10646] ISO/IEC 10646:2003. Information technology -- Universal Multiple-Octet Coded Character Set (UCS).

[CGI1.1] Robinson, D. and K. Coar, "The Common Gateway Interface (CGI) Version 1.1", RFC 3875, October 2004.

[HTML2.0] Berners-Lee, T. and D. Connolly, "Hypertext Markup Language - 2.0", RFC 1866, November 1995.

[MD2] Kaliski, B., "The MD2 Message-Digest Algorithm", RFC 1319, April 1992.

[MD4] Rivest, R., "The MD4 Message-Digest Algorithm", RFC 1320, April 1992.

[MD5] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[SHA1] Eastlake 3rd, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, September 2001.

[SSLv2] Hickman, Kipp, "The SSL Protocol", Netscape Communications Corp., Feb 9, 1995.

[SSLv3] Freier, A., Karlton, P., and P. Kocher, "The Secure Sockets Layer (SSL) Protocol Version 3.0", RFC 6101, August 2011.

[TLSv1.0] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[TLSv1.1] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[TLSv1.2] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[OPENSSL] OpenSSL Cryptography and SSL/TLS Toolkit at http://www.openssl.org/

[RFC1464] Rosenbaum, R., "Using the Domain Name System To Store Arbitrary String Attributes", RFC 1464, May 1993.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4279] Eronen, P., Ed., and H. Tschofenig, Ed., "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, December 2005.

[RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, May 2006.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC4785] Blumenthal, U. and P. Goel, "Pre-Shared Key (PSK) Ciphersuites with NULL Encryption for Transport Layer Security (TLS)", RFC 4785, January 2007.

[RFC5234] Crocker, D., Ed., and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5288] Salowey, J., Choudhury, A., and D. McGrew, "AES Galois Counter Mode (GCM) Cipher Suites for TLS", RFC 5288, August 2008.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, October 2008.

[RFC5469] Eronen, P., Ed., "DES and IDEA Cipher Suites for Transport Layer Security (TLS)", RFC 5469, February 2009.

[RFC5489] Badra, M. and I. Hajjeh, "ECDHE_PSK Cipher Suites for Transport Layer Security (TLS)", RFC 5489, March 2009.

[RFC5932] Kato, A., Kanda, M., and S. Kanno, "Camellia Cipher Suites for TLS", RFC 5932, June 2010.

[RFC6149] Turner, S. and L. Chen, "MD2 to Historic Status", RFC 6149, March 2011.

[RFC6150] Turner, S. and L. Chen, "MD4 to Historic Status", RFC 6150, March 2011.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, May 2011.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, August 2012.

[MYADDON] A working implementation of this Add-on on my website at https://ssl.mathemainzel.info/sslinfo/

9. Discussions

It would be good to have a standardized URL for this Add-on webpage; e.g. https://www.example.com/sslinfo/

Placing an Absolute URI as defined in [RFC3986] Section 4.3. outside the encrypted website part is RECOMMENDED.

A. Example certificates

A.1. The DER-encoded CA certificate

This section contains the full, DER-encoded certificate, in hex.

```
30820267308201D0A003020102020100300D06092A864886F70D010105050030
47310B300906035504061302D2D3110300E060355040A1307536F6D654F7267
31143012060355040B130B536F6D654F7267556E69743110300E060355040313
07526F6F74204341301E170D373030313031303030303030305A170D3734313233
313233353935395A3047310B300906035504061302D2D3110300E060355040A
1307536F6D654F726731143012060355040B130B536F6D654F7267556E697431
10300E060355040313075265F6F7420434130819F300D06092A864886F70D0101
01050003818D0030818902818100CE72969A54332263FBC26310BBEDA8EA0DC2
E0532C899CB314A1451D3A5A7CBB2ADCAF463449B2D1C6A2BC772285DF17FB12
067613CF3328459D7D7C4D847CF51C0F9562F296EFFB399C03CBE888FFBE4C11
57E032D88C8E87BF90A507F3D5DDD06E2A5FF19B4D2B89DF732DA7CBEA034C90
A4F1FEF58240943FD25793794E770203010001A3633061300F0603551D130101
FF040530030101FF300E0603551D0F0101FF040403020106301D0603551D0E04
16041473311472F1D56473C8DE0D0E39CCC2792B71EDDE301F0603551D230418
3016801473311472F1D56473C8DE0D0E39CCC2792B71EDDE300D06092A864886
F70D0101050500038181000C446885FF2B8451B00E668D530493474524E8EDE1
3B1AC325E677D9BE92204BA13369AFC48445AF3E01359B6C054D1049028DBC7A
E2F8A68BF5DCB89010C488B41896EB34C7B1DA195B2B7C26579CC2F7A705C4AE
9C4F72D80E8E3DD7AEC7B3154927B7FF8410712C9330E3FA98A5949A283CD599
FC8D9D97E330E05086844C
```

A.1.1. The CA’s public key

This section contains the DER-encoded public RSA key of the CA who signed the example SSL certificate. It is included with the purpose of simplifying verifications of the example certificate.

```
30819F300D06092A864886F70D010101050003818D0030818902818100CE7296
9A54332263FBC26310BBEDA8EA0DC2E0532C899CB314A1451D3A5A7CBB2ADCAF
463449B2D1C6A2BC772285DF17FB12067613CF3328459D7D7C4D847CF51C0F95
62F296EFFB399C03CBE888FFBE4C1157E032D88C8E87BF90A507F3D5DDD06E2A
5FF19B4D2B89DF732DA7CBEA034C90A4F1FEF58240943FD25793794E77020301
0001
```

A.2. The DER-encoded SSL certificate

This section contains the full, DER-encoded certificate, in hex.

```
30820289308201F2A003020102020101300D06092A864886F70D010105050030
47310B3009060355040613022D2D3110300E060355040A1307536F6D654F7267
31143012060355040B130B536F6D654F7267556E69743110300E060355040313
07526F6F74204341301E170D3730303130313030303030305A170D3730313233
313233353935395A3027310B3009060355040613022D2D311830160603550403
130F7777772E6578616D706C652E636F6D30819F300D06092A864886F70D0101
01050003818D0030818902818100D311D25BEDCC2765D1BF6BE9AB43C2ED41B
A9AF9531544186940E28AA5C80B460FED2EE1ABD5BE2BD6E351EAF9F0DCE4388
27B42E166FAE594C83F40B72175EE875342E3450FAF1407A12267E85041C94F9
A6DBBDC6F593958D0204199AE457EAB87D7E85487123C73398156F2AF1B87C49
0EF27B20F93A81C0165F6BCBEEE10203010001A381A43081A130090603551D13
04023000300E0603551D0F0101FF0404030205A0301D0603551D0E0416041454
64A24E922027FB19CFF91E7ECF846A0D50F2DC301F0603551D23041830168014
73311472F1D56473C8DE0D0E39CCC2792B71EDDE301A0603551D110413301182
0F7777772E6578616D706C652E636F6D30130603551D25040C300A06082B0601
05050703013013060355 1D20040C300A3008060667810C010201300D06092A86
4886F70D0101050500038181008A7F7627D29390ED474D591F2F4C94FCFCAEFA
DB04CBFD0619678A6001B1BC19CFD29AE96D48949DA81D1BCFE8F5E764BA7F91
C52BC50C28A472C2A6B2FEF4EB27BEE6B0C989AF1B7CF8E3A52F641B77C34E2A
7FB9E4B555F3843C592E0EE9C46DD9EABACBC915EE6D92E3C542C93739A6DBFE
637EB2B082566FBC46A3A60D46
```

B. Script Examples for the Add-on webpage

Use the following script examples as a template for your implementation of this Add-on webpage.

The first two examples generate identical content in plain ASCII-text, the third example makes use of HTML and is a compiled C program.

Script Examples:

B.1. PHP-script
B.2. CGI-script: A BASH shell script, for most Linux systems
B.3. CGI-script: A compiled C program, for any other system

B.1. PHP-script

```
<CODE BEGINS>
<?php

header( "Content-type: text/plain" );

print "SSL informations: " . date( "r" ) . "\r\n";
print "================\r\n\r\n";

if ( isset( $_SERVER['HTTPS'] ) &&
     ( $_SERVER['HTTPS'] == "on" ) ) {
  $list = array( );
  $nmbrOfValues = 0;

  foreach ( $_SERVER as $key => $value ) {
    if ( substr( $key, 0, 4 ) == "SSL_" ) {
      $list[ $nmbrOfValues++ ] = $key . "=" . $value;
    }
  }

  sort( $list );   // sort content before printing ...

  for ( $iter = 0; $iter < $nmbrOfValues; $iter++ ) {
    print $list[ $iter ] . "\r\n";
  }
}
else {
  echo "No SSL information available.\r\n";
}
?>
<CODE ENDS>
```

B.2. CGI-script: A BASH shell script, for most Linux systems

```
<CODE BEGINS>
#!/bin/bash

echo -e -n "Content-type: text/plain\n\n"

echo -e -n "SSL informations: $(date --rfc-2822)\n"
echo -e -n "================\n\n"

if [ "$HTTPS" == "on" ]; then
  env | grep --regexp="^SSL_" | sort
else
  echo -e -n "No SSL information available.\n"
fi
<CODE ENDS>
```

B.3. CGI-script: A compiled C program, for any other system

This CGI-script is a compiled C program, and in comparison to the other 2 examples, it makes use of HTML.

For compiling this program any C compiler SHOULD be suitable. Be sure your runtime supports the function strftime with standard format specifiers.

```
<CODE BEGINS>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#ifdef __linux__
#include <unistd.h>
#endif

const char* pszHtmlEndPart [ ] = { "<HR>",
  "<ADDRESS>https at www.example.com Port 443</ADDRESS>",
  "</BODY>",
  "</HTML>" };

const char* pszHtmlBeginPart[ ] = {
  "<!DOCTYPE HTML PUBLIC \"-//IETF//DTD HTML 2.0//EN\">",
  "<HTML>",
  "<HEAD>",
  "<TITLE>SSL informations</TITLE>",
  "</HEAD>",
```

```
    "<BODY>",
    "<H3>SSL informations</H3>" };

  /* function prototype used for sorting */
  int compareFunc( const void* pvd1, const void* pvd2 );

  int main( int argc, char* argv[ ], char** envp )
  {                              /* char* envp[ ] */
    char* * ppszContent;
    char* * ppsz;
    char* psz;
    char szDateTime[ 80 ];
    int i, nCount;

    time_t tnow = time( NULL );
    struct tm* tmnow = localtime( &tnow );

    strftime( szDateTime, sizeof( szDateTime ) - 4,
      "%a, %d %b %Y %H:%M:%S %z", tmnow );

    printf( "Content-type: text/html; charset=ISO-8859-1\r\n\r\n" );

    nCount = sizeof( pszHtmlBeginPart ) / sizeof( char* );
    for ( i = 0; i < nCount; i++ )
      printf( "%s\r\n", pszHtmlBeginPart[ i ] );

    printf( "<B>SSL informations</B>: %s\r\n", szDateTime );
    printf( "<P>\r\n" );

    if ( ( psz = getenv( "HTTPS" ) ) && ( strcmp( psz, "on" ) == 0 ) )
    {
      /* count relevant values ... */
      ppsz = envp;
      nCount = 0;
      while ( ppsz && *ppsz )
      {
        if ( strncmp( *ppsz, "SSL_", 4 ) == 0 )
          nCount++;
        ppsz++;
      }

      /* allocate memory */
      ppszContent = (char* *) calloc( nCount, sizeof( char* ) );
```

```
      if ( ppszContent )
      {
        /* extract relevant values from environment ... */
        i = 0;
        ppsz = envp;
        while ( ppsz && *ppsz )
        {
          if ( strncmp( *ppsz, "SSL_", 4 ) == 0 )
            *( ppszContent + i++ ) = *ppsz;
          ppsz++;
        }

        /* sort content */
        qsort( (void*) ppszContent, nCount, sizeof( char* ),
          compareFunc );

        printf( "<CODE>\r\n" );

        /* output sorted content */
        for ( i = 0; i < nCount; i++ )
          printf( "%s<BR>\r\n", *( ppszContent + i ) );

        printf( "</CODE>\r\n" );

        /* free up memory */
        free( (void*) ppszContent );
      }
      else
        printf( "Internal error (unable to allocate memory).\r\n" );
    }
    else
      printf( "No SSL information available.\r\n" );

    nCount = sizeof( pszHtmlEndPart ) / sizeof( char* );
    for ( i = 0; i < nCount; i++ )
      printf( "%s\r\n", pszHtmlEndPart[ i ] );

    return 0;
  }

  /* comparison function for sorting */
  int compareFunc( const void* pvd1, const void* pvd2 )
  {
    return strcmp( *( (char* *) pvd1 ), *( (char* *) pvd2 ) );
  }
  <CODE ENDS>
```

C. Sample Content of the Add-on webpage

The first example shows a complete sample content in sorted order. The second example shows the client certificate part, in case client certificate authentication is used. The other two examples show only the part that may differ when the browser picks another cipher suite.

For meaning of the numbers in brackets of the examples see Section 2.2.1.

C.1. A complete sample content

```
SSL informations: Thu, 01 Jan 1970 00:00:00 +0000              (1)
================

SSL_CIPHER=AES256-SHA                                          (2)
SSL_CIPHER_ALGKEYSIZE=256                                      (4)
SSL_CIPHER_EXPORT=false                                        (6)
SSL_CIPHER_USEKEYSIZE=256                                      (3)
SSL_CLIENT_VERIFY=NONE                                         (16)
SSL_COMPRESS_METHOD=NULL                                       (17)
SSL_PROTOCOL=TLSv1                                             (5)
SSL_SECURE_RENEG=true                                          (7)
SSL_SERVER_A_KEY=rsaEncryption                                 (8)
SSL_SERVER_A_SIG=sha1WithRSAEncryption                         (9)
SSL_SERVER_I_DN=/C=--/O=SomeOrg/OU=SomeOrgUnit/CN=Root CA      (10)
SSL_SERVER_I_DN_C=--                                           (10)
SSL_SERVER_I_DN_CN=Root CA                                     (10)
SSL_SERVER_I_DN_O=SomeOrg                                      (10)
SSL_SERVER_I_DN_OU=SomeOrgUnit                                 (10)
SSL_SERVER_M_SERIAL=01                                         (12)
SSL_SERVER_M_VERSION=3                                         (13)
SSL_SERVER_S_DN=/C=--/CN=www.example.com                       (11)
SSL_SERVER_S_DN_C=--                                           (11)
SSL_SERVER_S_DN_CN=www.example.com                             (11)
SSL_SERVER_V_END=Dec 31 23:59:59 1970 GMT                      (15)
SSL_SERVER_V_START=Jan 01 00:00:00 1970 GMT                    (14)
SSL_SESSION_ID=0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF (27)
SSL_TLS_SNI=www.example.com                                    (28)
SSL_VERSION_INTERFACE=mod_ssl/2.2.15                           (29)
SSL_VERSION_LIBRARY=OpenSSL/1.0.0-fips                         (30)
```

C.1.1. ..., the client certificate part

```
   ...
   SSL_CLIENT_A_KEY=rsaEncryption                                    (18)
   SSL_CLIENT_A_SIG=sha1WithRSAEncryption                            (19)
   SSL_CLIENT_I_DN=/C=--/O=SomeOrg/OU=SomeOrgUnit/CN=Root CA         (20)
   SSL_CLIENT_I_DN_C=--                                              (20)
   SSL_CLIENT_I_DN_CN=Root CA                                        (20)
   SSL_CLIENT_I_DN_O=SomeOrg                                         (20)
   SSL_CLIENT_I_DN_OU=SomeOrgUnit                                    (20)
   SSL_CLIENT_M_SERIAL=02                                            (22)
   SSL_CLIENT_M_VERSION=3                                            (23)
   SSL_CLIENT_S_DN=/CN=Name/emailAddress=name@example.com            (21)
   SSL_CLIENT_S_DN_CN=Name                                           (21)
   SSL_CLIENT_S_DN_Email=name@example.com                            (21)
   SSL_CLIENT_VERIFY=SUCCESS                                         (16)
   SSL_CLIENT_V_END=Dec 31 23:59:59 1970 GMT                         (25)
   SSL_CLIENT_V_REMAIN=365                                           (26)
   SSL_CLIENT_V_START=Jan 01 00:00:00 1970 GMT                       (24)
   ...
```

C.2. Picking another cipher suite

```
   ...
   SSL_CIPHER=RC4-MD5
   SSL_CIPHER_ALGKEYSIZE=128
   SSL_CIPHER_EXPORT=false
   SSL_CIPHER_USEKEYSIZE=128
   ...
   SSL_PROTOCOL=SSLv3
   SSL_SECURE_RENEG=false
   ...
```

C.2.1. ..., and one more

```
   ...
   SSL_CIPHER=AES128-SHA256
   SSL_CIPHER_ALGKEYSIZE=128
   SSL_CIPHER_EXPORT=false
   SSL_CIPHER_USEKEYSIZE=128
   ...
   SSL_PROTOCOL=TLSv1.2
   SSL_SECURE_RENEG=true
   ...
```

Author's Address

Walter Hoehlhubmer
Lederergasse 47a
A-4020 Linz
Austria, EUROPE

EMail: walter.h@mathemainzel.info