

ASN.1 templating for fun and profit

William Robinet

2023-07-05

#pts23

About me

- ▶ CompSci studies, work in IT (Conostix S.A. – AS197692)
- ▶ SSLDump improvements (build system, JSON output, IPv6 & ja3(s) support, ...)

PEM file – an X.509 certificate

-----BEGIN CERTIFICATE-----

MIIFPzCCBCegAwIBAgISBN0YAh1Gh1UW5NyEQUR1P6NeMA0GCSqGSIb3DQEBCwUA
MDIx CzA JBgNVBAYTA1VTMRYwFAYDVQQKEw1MZXQncyBFbmNyeXBOMQswCQYDVQQD
[...]

/4bgeLYcnWuE2oydoR9Vr/PpKGZcefWrb5oNu5Ttuui6VwD2ed1M8IcVG+n0KFKH
bydJ2Ra5KFCmBmkFq8ZVi6zH1Ldvk8B5qIeq7VsRycPFsEUc9mjFI7g04vBaHkc
gPzk

-----END CERTIFICATE-----

Transport over text channels – PEM

- ▶ Privacy-Enhanced Mail
- ▶ IETF RFC 1421 and IETF RFC 7468
- ▶ Base64 + surrounding BEGIN/END tags
- ▶ Useful for transport of binary structures over text-based channels such as email

Transport over text channels – PEM (2)

Valid types (from openssl-format-options(1)):

RSA PUBLIC KEY	ECDSA PUBLIC KEY
X509 CERTIFICATE	ENCRYPTED PRIVATE KEY
DSA PRIVATE KEY	PARAMETERS
NEW CERTIFICATE REQUEST	PKCS #7 SIGNED DATA
ANY PRIVATE KEY	PKCS7
CERTIFICATE	PRIVATE KEY
CERTIFICATE REQUEST	PUBLIC KEY
CMS	RSA PRIVATE KEY
DH PARAMETERS	SSL SESSION PARAMETERS
DSA PARAMETERS	TRUSTED CERTIFICATE
DSA PUBLIC KEY	X509 CRL
EC PARAMETERS	X9.42 DH PARAMETERS
EC PRIVATE KEY	

Binary Encoded file

```
$ hexdump -C certificate.der
```

```
00000000 30 82 05 3f 30 82 04 27 a0 03 02 01 02 02 12 04 |0...?0...'.....|
00000010 dd 18 02 19 46 87 55 16 e4 dc 84 41 44 75 3f a3 |....F.U....ADu?.|
00000020 5e 30 0d 06 09 2a 86 48 86 f7 0d 01 01 0b 05 00 |^0...*.H.....|
00000030 30 32 31 0b 30 09 06 03 55 04 06 13 02 55 53 31 |021.0...U....US1|
00000040 16 30 14 06 03 55 04 0a 13 0d 4c 65 74 27 73 20 |.0...U....Let's |
00000050 45 6e 63 72 79 70 74 31 0b 30 09 06 03 55 04 03 |Encrypt1.0...U..|
00000060 13 02 52 33 30 1e 17 0d 32 33 30 36 30 33 30 36 |..R30...23060306|
00000070 35 37 33 35 5a 17 0d 32 33 30 39 30 31 30 36 35 |5735Z..230901065|
[...]
00000520 ac c7 94 b7 6f 93 c0 79 a8 87 9a ab b5 6c 47 27 |....o..y.....lG'|
00000530 0f 16 c1 14 73 d9 a3 14 8e e0 3b 8b c1 68 79 1c |....s.....;..hy.|
00000540 80 fc e4 |...|
00000543
```

Binary Encoded file (2)

```
$ file certificate.der  
certificate.der: data
```

Fixed in 2021¹

```
$ file certificate.der  
certificate.der: Certificate, Version=3
```

²<https://github.com/file/file/commit/0d6c87c6a63c91077b6f55334f31ec4ca545718f>

Binary Encoded file (3)

```
$ openssl x509 -text -noout -in certificate.der -inform D
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      04:dd:18:02:19:46:87:55:16:e4:dc:84:41:44:75:3f:a3:5e
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C = US, O = Let's Encrypt, CN = R3
    Validity
      Not Before: Jun  3 06:57:35 2023 GMT
      Not After : Sep  1 06:57:34 2023 GMT
    Subject: CN = pass-the-salt.org
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (2048 bit)
      Modulus:
        00:b1:34:03:ed:cb:34:13:4d:b0:d1:20:68:d1:ab:
[...]
```


Binary Encoded file (4)

```
$ man openssl-asn1parse
```

```
[...]
```

```
NAME
```

```
    openssl-asn1parse, asn1parse - ASN.1 parsing tool
```

```
[..]
```

```
DESCRIPTION
```

```
    The asn1parse command is a diagnostic utility that can parse  
    ASN.1 structures. It can also be used to extract data from ASN.1  
    formatted data.
```

```
[...]
```

Binary Encoded file (5)

```
$ openssl asn1parse -in certificate.der -inform D -i
 0:d=0  hl=4 l=1343 cons: SEQUENCE
 4:d=1  hl=4 l=1063 cons: SEQUENCE
 8:d=2  hl=2 l= 3 cons: cont [ 0 ]
10:d=3  hl=2 l= 1 prim: INTEGER           :02
13:d=2  hl=2 l= 18 prim: INTEGER           :04DD18021946875516E4DC844144753FA35E
33:d=2  hl=2 l= 13 cons: SEQUENCE
35:d=3  hl=2 l= 9 prim: OBJECT              :sha256WithRSAEncryption
46:d=3  hl=2 l= 0 prim: NULL
48:d=2  hl=2 l= 50 cons: SEQUENCE
50:d=3  hl=2 l= 11 cons: SET
52:d=4  hl=2 l= 9 cons: SEQUENCE
54:d=5  hl=2 l= 3 prim: OBJECT              :countryName
59:d=5  hl=2 l= 2 prim: PRINTABLESTRING    :US
63:d=3  hl=2 l= 22 cons: SET
65:d=4  hl=2 l= 20 cons: SEQUENCE
67:d=5  hl=2 l= 3 prim: OBJECT              :organizationName
72:d=5  hl=2 l= 13 prim: PRINTABLESTRING    :Let's Encrypt
[...]
```

ASN.1 Example – RSA private key

```
RSAPrivateKey ::= SEQUENCE {  
    version Version,  
    modulus INTEGER, -- n  
    publicExponent INTEGER, -- e  
    privateExponent INTEGER, -- d  
    prime1 INTEGER, -- p  
    prime2 INTEGER, -- q  
    exponent1 INTEGER, -- d mod (p-1)  
    exponent2 INTEGER, -- d mod (q-1)  
    coefficient INTEGER, -- (inverse of q) mod p  
    otherPrimeInfos OtherPrimeInfos OPTIONAL  
}  
  
Version ::= INTEGER  
  
OtherPrimeInfos ::= SEQUENCE OF OtherPrimeInfo  
  
OtherPrimeInfo ::= SEQUENCE {  
    prime INTEGER, -- ri  
    exponent INTEGER, -- di  
    coefficient INTEGER -- ti  
}
```

Abstract Syntax Notation One

- ▶ Standard interface description language (IDL)
- ▶ Used to define data structures that can be serialized and deserialized in a cross-platform way²
- ▶ ITU-T & ISO/IEC standard from the 80's
- ▶ Used in PKCS, X.500, LDAP, Kerberos, SNMP, EMV, GSM, UMTS, LTE, 5G, ...
- ▶ Our use case is the definition of cryptographic structures such as X.509 certificates, RSA/EC keys, ...
- ▶ Recommended reading:
`https://letsencrypt.org/docs/a-warm-welcome-to-asn1-and-der/`

¹Source: <https://en.wikipedia.org/wiki/ASN.1>

ASN.1 Tags

Basic types (primitive)

BOOLEAN

INTEGER

NULL

OID

UTCTIME, GENERALIZEDTIME

OCTET STRING

BIT STRING

UTF8String, PRINTABLESTRING, ...

ENUM

Structured types (constructed)

SEQUENCE, SEQUENCE OF

SET, SET OF

ASN.1 Tag Classes

- ▶ Specify encoding instructions in order to remove ambiguity where necessary
- ▶ It helps defining our own tags using a class and a tag value ($\text{uint} < 2^{31} - 1$)
- ▶ Available classes are:
UNIVERSAL, **APPLICATION**, **CONTEXT SPECIFIC** or **PRIVATE**
- ▶ Classes encoding can be **IMPLICIT** or **EXPLICIT**

Binary Encodings

- ▶ **BER – Basic Encoding Rules**
- ▶ **DER – Distinguished Encoding Rules**
- ▶ CER – Canonical Encoding Rules
- ▶ PER – Basic Packed Encoding Rules (Aligned/Unaligned)
- ▶ CPER – Canonical Packed Encoding Rules (Aligned/Unaligned)
- ▶ XER – Basic XML Encoding Rules
- ▶ CXER – Canonical XML Encoding Rules
- ▶ EXER – Extended XML Encoding Rules
- ▶ OER – Octet Encoding Rules
- ▶ JER – JSON Encoding Rules
- ▶ GSER – Generic String Encoding Rules
- ▶ SER – Signalling Specific Encoding Rules
- ▶ LWER – Lightweight Encoding Rules
- ▶ MBER – Minimum Bit Encoding Rules

BER/DER structures encodings

BER – Basic Encoding Rules

DER – Distinguished Encoding Rules

- ▶ Byte based binary formats
- ▶ TLV – Type/Tag, Length, Value
- ▶ BER allows encoding of a given ASN.1 structure in multiple ways
- ▶ BER is useful for streams (content not known in advance)
- ▶ DER is a subset of BER along with canonicalization rules
- ▶ DER only allows a single encoding for a given ASN.1 structure
- ▶ That's why it is used for signed data structures (fixed content)

ASN.1/DER Tags encoding

Tag value (hex)	Tag
02	INTEGER
03	BIT STRING
04	OCTET STRING
05	NULL
06	OBJECT IDENTIFIER
0C	UTF8String
10(Constructed → 30)	SEQUENCE and SEQUENCE OF
11(Constructed → 31)	SET and SET OF
13	PrintableString
16	IA5String
17	UTCTime
18	GeneralizedTime

Bit #6 indicates a Constructed tag compared to a Primitive tag

ASN.1/DER Tag Class encoding

Bits #8 & #7 are use for class encoding

Class	Bit #8	Bit #7
Universal	0	0
Application	0	1
Context-specific	1	0
Private	1	1

DER encoding examples (2)

```
my_struct ::= SEQUENCE {  
    int0 INTEGER:0x12  
    int1 INTEGER:0x34  
}
```

```
/--> type: SEQUENCE  
/ /--> length: 6 bytes  
/ / //////////////////////////////////--> value: the two DER encoded INTEGERS  
30 06 02 01 12 02 01 34  
  / / / / / /--> value: 0x34  
  / / / / /--> length: 1 byte  
  / / / /--> type: INTEGER  
  / / /--> value: 0x12  
  / /--> length: 1 byte  
  /--> type: INTEGER
```

DER encoding examples (2)

```
my_struct ::= SEQUENCE {  
    int0 INTEGER:0x12  
    int1 INTEGER:0x34  
}
```

```
/--> type: SEQUENCE  
/ /--> length: 6 bytes  
/ / //////////////////////////////////////////////////--> value: the two DER encoded INTEGERS  
30 06 02 01 12 02 01 34  
  / / / / / /--> value: 0x34  
  / / / / /--> length: 1 byte  
  / / / /--> type: INTEGER  
  / / /--> value: 0x12  
  / /--> length: 1 byte  
  /--> type: INTEGER
```

How do we edit these kind of structures ?

Motivation, why would you want to do this ?

- ▶ Exploiting known vulnerabilities
- ▶ Testing the limits ! (fuzzing)

DER editing manual example

SEQUENCE:

INTEGER:0x12

30 06 02 01 12 02 01 34

INTEGER:0x34

SEQUENCE:

INTEGER:0x12

INTEGER:0x3456

/--> Outer SEQUENCE is now 7 bytes long

/

/--> second INTEGER in outer SEQUENCE is now

/

/

2 bytes long

30 07 02 01 12 02 02 34 56

DER editing manual example

SEQUENCE:

INTEGER:0x12

30 06 02 01 12 02 01 34

INTEGER:0x34

SEQUENCE:

INTEGER:0x12

INTEGER:0x3456

/--> Outer SEQUENCE is now 7 bytes long

/

/--> second INTEGER in outer SEQUENCE is now

/

/

2 bytes long

30 07 02 01 12 02 02 34 56

DER editing manual example

SEQUENCE:

INTEGER:0x12

30 06 02 01 12 02 01 34

INTEGER:0x34

SEQUENCE:

INTEGER:0x12

INTEGER:0x3456

/--> Outer SEQUENCE is now 7 bytes long

/

/--> second INTEGER in outer SEQUENCE is now

/

/

2 bytes long

30 07 02 01 12 02 02 34 56

DER editing manual example (2)

Larger structures composed of multiple depth of nested sub-structures are a pain to edit.

[illegible]

A solution



The “asn1template.pl” script – Idea

```
$ man openssl-asn1parse
```

```
[...]
```

```
-genstr string, -genconf file
```

Generate encoded data based on string, file or both using ASN1_generate_nconf(3) format. If file only is present then the string is obtained from the default section using the name asn1. The encoded data is passed through the ASN1 parser and printed out as though it came from a file, the contents can thus be examined and written to a file using the out option.

```
[...]
```

The “asn1template.pl” script – How it works

Internal structure of the script:

- ▶ Read the output of the “asn1parse” OpenSSL app
(a **representation** of the tree structure)
- ▶ Reconstruct the ASN.1 structure tree
- ▶ Traverse the tree recursively, depth-first
- ▶ and write the “-genconf” compatible output “ASN1_generate_nconf(3)”
(a **description** of the tree structure)
- ▶ First version written around 2010. “der-ascii” did not exist back then.
- ▶ Written in PERL, depends on the OpenSSL CLI utility

asn1parse – representation

```
$ openssl asn1parse -in test.der -i -inform D
  0:d=0  hl=2 l= 18 cons: SEQUENCE
  2:d=1  hl=2 l=  8 cons:  SEQUENCE
  4:d=2  hl=2 l=  6 cons:   SEQUENCE
  6:d=3  hl=2 l=  4 prim:    INTEGER           :76543210
 12:d=1  hl=2 l=  6 cons:  SEQUENCE
 14:d=2  hl=2 l=  4 prim:    INTEGER           :01234567
```

genconf option format – description

```
$ asn1template.pl test.der
asn1 = SEQUENCE:seq1@0-2-18
[seq1@0-2-18]
field2@2-2-8 = SEQUENCE:seq2@2-2-8
field3@12-2-6 = SEQUENCE:seq3@12-2-6
[seq2@2-2-8]
field4@4-2-6 = SEQUENCE:seq4@4-2-6
[seq4@4-2-6]
field5@6-2-4 = INTEGER:0x76543210
[seq3@12-2-6]
field6@14-2-4 = INTEGER:0x01234567
```

Demos: (More or less) recent DoS vulnerabilities in OpenSSL

- ▶ CVE-2022-0778 (<https://www.openssl.org/news/secadv/20220315.txt>)
 - Infinite loop in BN_mod_sqrt() reachable when parsing certificates
 - Edition of a certificate with the broken EC key parameters
- ▶ CVE-2023-2650 (<https://www.openssl.org/news/secadv/20230530.txt>)
 - Possible DoS translating ASN.1 object identifiers
 - Add a ridiculously long OID to an existing structure

Alternative SSL/TLS tool suites

- ▶ Works with LibreSSL out of the box
- ▶ BoringSSL: genconf option not available, needs an additional helper tool
- ▶ ...

Improvements & future evolutions

- ▶ Fix IMPLICIT/EXPLICIT tagging
- ▶ Support missing data types
- ▶ Better encoding detection
- ▶ Support for binary fixers (indefinite length, out of standard values, ...)
- ▶ PR welcome :)

Contact

project page

<https://www.github.com/wllm-rbnt/asn1template>

social media

@wr@infosec.exchange

email

willi@mrobi.net

slides

<https://www.github.com/wllm-rbnt/asn1template/pts2023/asn1template.pdf>³

³Built & presented on Qubes-OS :)