Trapped by the CLI

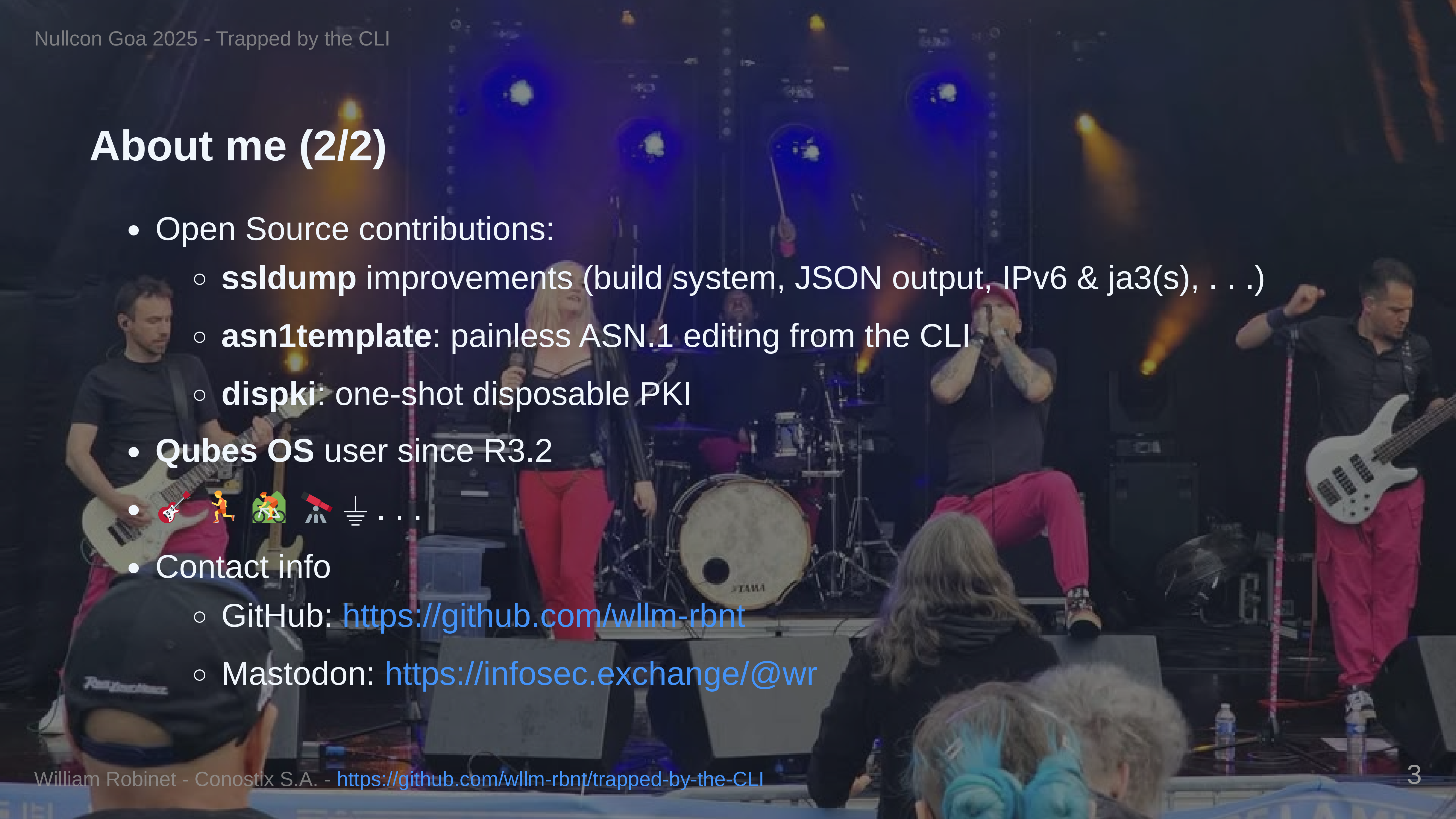
How your favorite cryptographic toolkit can fool you

Nullcon Goa 2025

William Robinet - Conostix S.A. - https://github.com/wllm-rbnt/trapped-by-the-CLI

About me (1/2)

- Introduced to Open Source & Free Software around the end of the 90's
- CompSci studies
- CTO at Conostix S.A. Luxembourg - AS197692
- **LUNOG** founding member
- Member of the **BSides Luxembourg** revival team
- Conference workshops & talks



About me (2/2)

- Open Source contributions:
  - **ssldump** improvements (build system, JSON output, IPv6 & ja3(s), . . .)
  - **asn1template**: painless ASN.1 editing from the CLI
  - **dispki**: one-shot disposable PKI
- **Qubes OS** user since R3.2
- 🎸 🏃 🚵 🔭 ⏚ . . .
- Contact info
  - GitHub: https://github.com/wllm-rbnt
  - Mastodon: https://infosec.exchange/@wr

Outline

- X.509, PEM, DER & ASN.1 overview
- CVE-2022-0778 & cryptographic data structures editing
- The story behind this talk
- Experiments & findings
- Real world implications

X.509, PEM, DER & ASN.1 overview

An X.509 certificate is an ASN.1 structure usually encoded as a DER binary object maybe encapsulated in PEM format

PEM format ? DER binary object ?? ASN.1 structure ??!?!

An X.509 certificate printed via the openssl x509 command

```
$ echo | openssl s_client -connect google.com:443 | openssl x509 -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            df:5b:69:1b:21:76:4a:32:12:1c:2f:37:8b:e5:96:d3
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, O = Google Trust Services, CN = WR2
        Validity
            Not Before: Jan 27 08:35:27 2025 GMT
            Not After : Apr 21 08:35:26 2025 GMT
        Subject: CN = *.google.com
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:a0:26:86:48:d6:05:d9:37:36:b3:e3:d0:44:3e:
                    [...]
                    7b:85:b9:a8:7f
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
[...]
```

An X.509 certificate in PEM format

```
$ echo | openssl s_client -connect google.com:443 | openssl x509
-----BEGIN CERTIFICATE-----
MIIOCjCCDPKgAwIBAgIRAN9baRshdkoyEhwvN4vlltMwDQYJKoZIhvcNAQELBQAw
OzELMAkGA1UEBhMCVVMxHjAcBgNVBAoTFUdvb2dsZSBUcnVzdCBTZXJ2aWNlczEM
MAoGA1UEAxMDV1IyMB4XDTI1MDEyNzA4MzUyN1oXDTI1MDQyMTA4MzUyNlowFzEV
MBMGA1UEAwwMKi5nb29nbGUuY29tMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE
oCaGSNYF2Tc2s+PQRD5kX2GInK49w1IBRF6Rf97ixqizgxIfA4GtZJZ5gAjk0SWH
gORZ+CvgnMUbyix7hbmof6OCC/YwggvyMA4GA1UdDwEB/wQEAwIHgDATBgNVHSUE
DDAKBggrBgEFBQcDATAMBgNVHRMBAf8EAjAAMB0GA1UdDgQWBBTmdR9tqDky5r2V
7LHRlNvVU4H4FDAfBgNVHSMEGDAWgBTeGx7teRXUPjckwyG77DQ5bUKyMDBYBggr
BgEFBQcBAQRMMEowIQYIKwYBBQUHMAGGFWh0dHA6Ly9vLnBraS5nb29nL3dyMjAl
BggrBgEFBQcwAoYZaHR0cDovL2kucGtpLmdvb2cvd3IyLmNydDCCCc0GA1UdEQSC
CcQwggnAggwqLmdvb2dsZS5jb22CFiouYXBwZW5naW5lLmdvb2dsZS5jb22CCSou
YmRuLmRldoIVKi5vcmlnaW4tdGVzdC5iZG4uZGV2ghIqLmNsb3VkLmdvb2dsZS5j
b22CGCouY3Jvd2Rzb3VyY2UuZ29vZ2xlLmNvbYIYKi5kYXRhY29tcHV0ZS5nb29n
bGUuY29tggsqLmdvb2dsZS5jYYILKi5nb29nbGUuY2yCDiouZ29vZ2xlLmNvLmlu
gg4qLmdvb2dsZS5jby5qcIIOKi5nb29nbGUuY28udWuCDyouZ29vZ2xlLmNvbS5h
[...]
6bVuRg8RIdWQi7fpCgqioGswTqeLIKzaALCxwKhXy4PaXLeU4FvEntM4n8Awbuk
0YOCzDtSjiFal8rSnOGbaorSymegOcfPhhrzm2UmPLUKEh2V+2BNi/9K2AdL/g==
-----END CERTIFICATE-----
```

The PEM format

```
-----BEGIN CERTIFICATE-----
MIIOCjCCDPKgAwIBAgIRAN9baRshdkoyEhwvN4vlltMwDQYJKoZIhvcNAQELBQAw
OzELMAkGA1UEBhMCVVMxHjAcBgNVBAoTFUdvb2dsZSBUcnVzdCBTZXJ2aWNlczEM
MAoGA1UEAxMDV1IyMB4XDTI1MDEyNzA4MzUyN1oXDTI1MDQyMTA4MzUyNlowFzEV
MBMGA1UEAwwMKi5nb29nbGUuY29tMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE
oCaGSNYF2Tc2s+PQRD5kX2GInK49w1IBRF6Rf97ixqizgxIfA4GtZJZ5gAjk0SWH
gORZ+CvgnMUbyix7hbmof6OCC/YwggvyMA4GA1UdDwEB/wQEAwIHgDATBgNVHSUE
DDAKBggrBgEFBQcDATAMBgNVHRMBAf8EAjAAMB0GA1UdDgQWBBTmdR9tqDky5r2V
7LHRlNvVU4H4FDAfBgNVHSMEGDAWgBTeGx7teRXUPjckwyG77DQ5bUKyMDBYBggr
BgEFBQcBAQRMMEowIQYIKwYBBQUHMAGGFWh0dHA6Ly9vLnBraS5nb29nL3dyMjAl
[...]
/4bgeLYcnWuE2oydoR9Vr/PpKGZcefWrb5oNu5Ttuui6VwD2edlM8IcVG+nOKFKH
bydJ2Ra5KFCmBmkFq8ZVi6zHlLdvk8B5qIeaq7VsRycPFsEUc9mjFI7gO4vBaHkc
gPzk
-----END CERTIFICATE-----
```

- 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

The PEM format - Transport over text channels

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
```

DER binary encoding (1) - binary output

```
$ echo | openssl s_client -connect google.com:443 | openssl x509 -outform DER | hexdump -C
00000000  30 82 0e 0a 30 82 0c f2  a0 03 02 01 02 02 11 00  |0...0...........|
00000010  df 5b 69 1b 21 76 4a 32  12 1c 2f 37 8b e5 96 d3  |.[i.!vJ2../7....|
00000020  30 0d 06 09 2a 86 48 86  f7 0d 01 01 0b 05 00 30  |0...*.H........0|
00000030  3b 31 0b 30 09 06 03 55  04 06 13 02 55 53 31 1e  |;1.0...U....US1.|
00000040  30 1c 06 03 55 04 0a 13  15 47 6f 6f 67 6c 65 20  |0...U....Google |
00000050  54 72 75 73 74 20 53 65  72 76 69 63 65 73 31 0c  |Trust Services1.|
00000060  30 0a 06 03 55 04 03 13  03 57 52 32 30 1e 17 0d  |0...U....WR20...|
00000070  32 35 30 31 32 37 30 38  33 35 32 37 5a 17 0d 32  |250127083527Z..2|
00000080  35 30 34 32 31 30 38 33  35 32 36 5a 30 17 31 15  |50421083526Z0.1.|
00000090  30 13 06 03 55 04 03 0c  0c 2a 2e 67 6f 6f 67 6c  |0...U....*.googl|
000000a0  65 2e 63 6f 6d 30 59 30  13 06 07 2a 86 48 ce 3d  |e.com0Y0...*.H.=|
000000b0  02 01 06 08 2a 86 48 ce  3d 03 01 07 03 42 00 04  |....*.H.=....B..|
000000c0  a0 26 86 48 d6 05 d9 37  36 b3 e3 d0 44 3e 64 5f  |.&.H...76...D>d_|
000000d0  61 88 9c ae 3d c3 52 01  44 5e 91 7f de e2 c6 a8  |a...=.R.D^......|
000000e0  b3 83 12 1f 03 81 ad 64  96 79 80 08 e4 d1 25 87  |.......d.y....%.|
000000f0  80 e4 59 f8 2b e0 9c c5  1b ca 2c 7b 85 b9 a8 7f  |..Y.+.....,{....|
[...]
00000df0  8a d2 ca 67 a0 39 c7 cf  86 1a f3 9b 65 26 3c b5  |...g.9......e&<.|
00000e00  0a 12 1d 95 fb 60 4d 8b  ff 4a d8 07 4b fe        |.....`M..J..K.|
00000e0e
```

DER binary encoding (2) - openssl x509 command

```
$ echo | openssl s_client -connect google.com:443 | openssl x509 -outform d | openssl x509 -inform d -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            df:5b:69:1b:21:76:4a:32:12:1c:2f:37:8b:e5:96:d3
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, O = Google Trust Services, CN = WR2
        Validity
            Not Before: Jan 27 08:35:27 2025 GMT
            Not After : Apr 21 08:35:26 2025 GMT
        Subject: CN = *.google.com
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:a0:26:86:48:d6:05:d9:37:36:b3:e3:d0:44:3e:
                    [...]
                    7b:85:b9:a8:7f
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature
[...]
```

DER binary encoding (3) - openssl asn1parse command

```
$ 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.
[...]
```

DER binary encoding (4) - openssl asn1parse command

```
$ echo | openssl s_client -connect google.com:443 | openssl x509 -outform d | openssl asn1parse -inform d -i
    0:d=0  hl=4 l=3594 cons: SEQUENCE
    4:d=1  hl=4 l=3314 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=  17 prim:   INTEGER           :DF5B691B21764A32121C2F378BE596D3
   32:d=2  hl=2 l=  13 cons:   SEQUENCE
   34:d=3  hl=2 l=   9 prim:    OBJECT            :sha256WithRSAEncryption
   45:d=3  hl=2 l=   0 prim:    NULL
   47:d=2  hl=2 l=  59 cons:   SEQUENCE
   49:d=3  hl=2 l=  11 cons:    SET
   51:d=4  hl=2 l=   9 cons:     SEQUENCE
   53:d=5  hl=2 l=   3 prim:      OBJECT            :countryName
   58:d=5  hl=2 l=   2 prim:      PRINTABLESTRING   :US
   62:d=3  hl=2 l=  30 cons:    SET
   64:d=4  hl=2 l=  28 cons:     SEQUENCE
   66:d=5  hl=2 l=   3 prim:      OBJECT            :organizationName
   71:d=5  hl=2 l=  21 prim:      PRINTABLESTRING   :Google Trust Services
   94:d=3  hl=2 l=  12 cons:    SET
   96:d=4  hl=2 l=  10 cons:     SEQUENCE
   98:d=5  hl=2 l=   3 prim:      OBJECT            :commonName
  103:d=5  hl=2 l=   3 prim:      PRINTABLESTRING   :WR2
[...]
```

Binary Encodings for ASN.1

- BER – Basic Encoding Rules
- DER – Distinguished Encoding Rules
- CER – Canonical Encoding Rules
- XER – Basic XML Encoding Rules
- CXER – Canonical XML Encoding Rules
- EXER – Extended XML Encoding Rules
- OER – Octet Encoding Rules
- JER – JSON Encoding Rules

. . .

ASN.1 Example – (grossly simplified) X.509 Certificate

```
Certificate  ::=  SEQUENCE  {
     tbsCertificate       TBSCertificate,
     signatureAlgorithm   AlgorithmIdentifier,
     signature            BIT STRING
}

TBSCertificate ::= SEQUENCE {
    version INTEGER,
    serialNumber INTEGER,
    signature AlgorithmIdentifier,
    issuer SEQUENCE of PrintableString,
    SEQUENCE {
        notBefore      UTCTime,
        notAfter       UTCTime
    }
    subject SEQUENCE of PrintableString,
    subjectPublicKeyInfo SubjectPublicKeyInfo,
    [...]
}
```

Reference: https://www.rfc-editor.org/rfc/rfc2459#page-15

Abstract Syntax Notation One - ASN.1

- Interface Description Language (IDL)
- From https://en.wikipedia.org/wiki/ASN.1 : used to define data structures that can be serialized and deserialized in a cross-platform way
- 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 data structures such as X.509 certificates, RSA/EC keys, . . .
- Recommended reading: https://letsencrypt.org/docs/a-warm-welcome-to-asn1-and-der/

ASN.1 Tags (serve as Types Identifiers)

Basic types (primitive)

BOOLEAN

INTEGER

NULL

OID

UTCTIME, GENERALIZEDTIME

OCTET STRING

BIT STRING

UTF8String, PRINTABLESTRING, . . .

ENUM

Structured types (constructed)

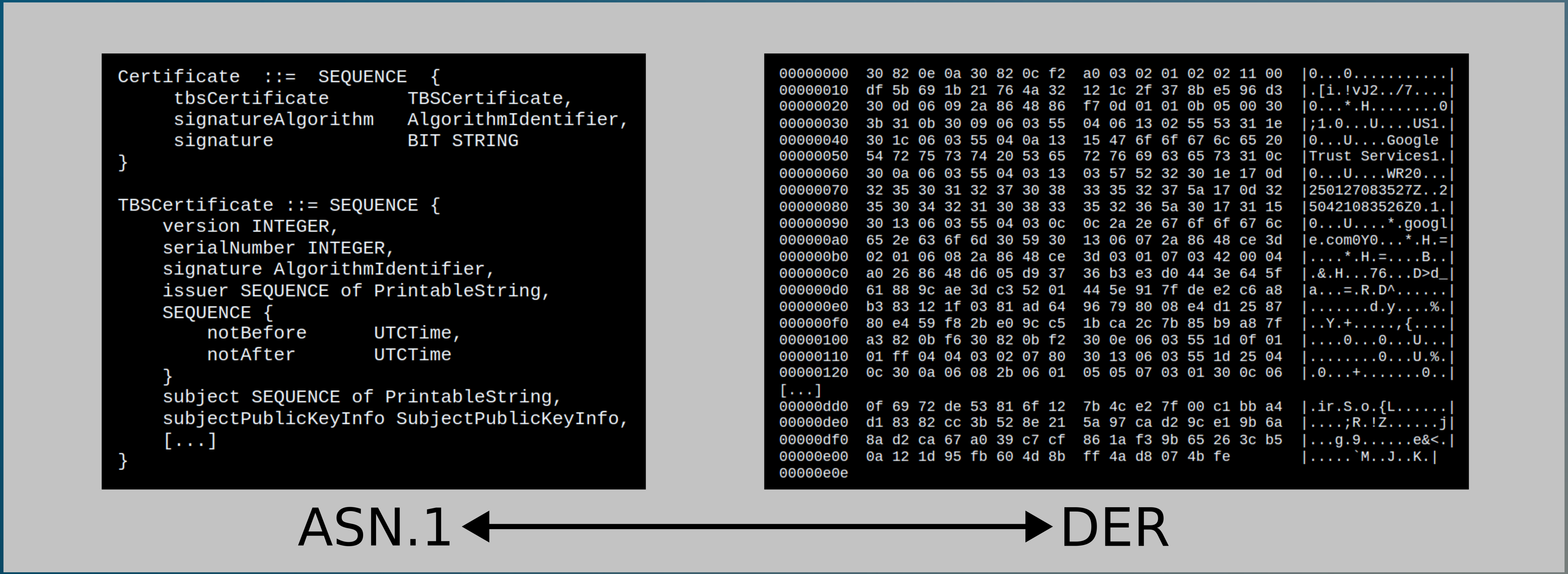
SEQUENCE, SEQUENCE OF

SET, SET OF

ASN.1 Tag Classes (give Context Definition to Tags)

- Specify encoding instructions in order to remove ambiguity where necessary
- It helps defining our own tags using a class and a tag value
- Tags can be encoded using 1 byte, up to 6 bytes (uint $< 2^{31} - 1$ max)
- Available classes are:
  UNIVERSAL (default), APPLICATION, CONTEXT SPECIFIC or PRIVATE
- Classes encoding can be IMPLICIT or EXPLICIT

DER Encoded X.509 certificate



```
Certificate  ::=  SEQUENCE  {
     tbsCertificate       TBSCertificate,
     signatureAlgorithm   AlgorithmIdentifier,
     signature            BIT STRING
}

TBSCertificate ::= SEQUENCE {
    version INTEGER,
    serialNumber INTEGER,
    signature AlgorithmIdentifier,
    issuer SEQUENCE of PrintableString,
    SEQUENCE {
        notBefore      UTCTime,
        notAfter       UTCTime
    }
    subject SEQUENCE of PrintableString,
    subjectPublicKeyInfo SubjectPublicKeyInfo,
    [...]
}
```

```
00000000  30 82 0e 0a 30 82 0c f2  a0 03 02 01 02 02 11 00  |0...0...........|
00000010  df 5b 69 1b 21 76 4a 32  12 1c 2f 37 8b e5 96 d3  |.[i.!vJ2../7....|
00000020  30 0d 06 09 2a 86 48 86  f7 0d 01 01 0b 05 00 30  |0...*.H........0|
00000030  3b 31 0b 30 09 06 03 55  04 06 13 02 55 53 31 1e  |;1.0...U....US1.|
00000040  30 1c 06 03 55 04 0a 13  15 47 6f 6f 67 6c 65 20  |0...U....Google |
00000050  54 72 75 73 74 20 53 65  72 76 69 63 65 73 31 0c  |Trust Services1.|
00000060  30 0a 06 03 55 04 03 13  03 57 52 32 30 1e 17 0d  |0...U....WR20...|
00000070  32 35 30 31 32 37 30 38  33 35 32 37 5a 17 0d 32  |250127083527Z..2|
00000080  35 30 34 32 31 30 38 33  35 32 36 5a 30 17 31 15  |50421083526Z0.1.|
00000090  30 13 06 03 55 04 03 0c  0c 2a 2e 67 6f 6f 67 6c  |0...U....*.googl|
000000a0  65 2e 63 6f 6d 30 59 30  13 06 07 2a 86 48 ce 3d  |e.com0Y0...*.H.=|
000000b0  02 01 06 08 2a 86 48 ce  3d 03 01 07 03 42 00 04  |....*.H.=....B..|
000000c0  a0 26 86 48 d6 05 d9 37  36 b3 e3 d0 44 3e 64 5f  |.&.H...76...D>d_|
000000d0  61 88 9c ae 3d c3 52 01  44 5e 91 7f de e2 c6 a8  |a...=.R.D^......|
000000e0  b3 83 12 1f 03 81 ad 64  96 79 80 08 e4 d1 25 87  |.......d.y....%.|
000000f0  80 e4 59 f8 2b e0 9c c5  1b ca 2c 7b 85 b9 a8 7f  |..Y.+.....,{....|
00000100  a3 82 0b f6 30 82 0b f2  30 0e 06 03 55 1d 0f 01  |....0...0...U...|
00000110  01 ff 04 04 03 02 07 80  30 13 06 03 55 1d 25 04  |........0...U.%.|
00000120  0c 30 0a 06 08 2b 06 01  05 05 07 03 01 30 0c 06  |.0...+.......0..|
[...]
00000dd0  0f 69 72 de 53 81 6f 12  7b 4c e2 7f 00 c1 bb a4  |.ir.S.o.{L......|
00000de0  d1 83 82 cc 3b 52 8e 21  5a 97 ca d2 9c e1 9b 6a  |....;R.!Z......j|
00000df0  8a d2 ca 67 a0 39 c7 cf  86 1a f3 9b 65 26 3c b5  |...g.9......e&<.|
00000e00  0a 12 1d 95 fb 60 4d 8b  ff 4a d8 07 4b fe        |.....`M..J..K.|
00000e0e
```

BER/DER structure encodings

- Byte based binary formats
- **TLV** – Type(aka 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)

DER Tag & Class encoding (first byte)

Tag (hex byte)	Tag name
02	INTEGER
03	BIT STRING
0C	UTF8String
10 (Constructed : 30)	SEQUENCE and SEQUENCE OF
13	PrintableString
17	UTCTime
. . .	. . .

Class	Bit #7	Bit #6
Universal	0	0
Application	0	1
Context-specific	1	0
Private	1	1

IMPLICIT class encoding

```
      /--/--> Tag Class
      /  /
Bit #7 #6 #5 #4 #3 #2 #1 #0
           \  \--\--\--\--\--> Tag
           \
           \--> Primitive or Constructed
```

- Bit #5 indicates a Constructed tag Vs a Primitive tag
- Bits #8 & #7 are use for class encoding

EXPLICIT class encoding

- The tag in embedded into a surrounding SEQUENCE that holds the class encoding

DER TLV encoding example

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

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

Parsing an encoded X.509 certificate with openssl asn1parse

```
$ echo | openssl s_client -connect google.com:443 | openssl x509 | openssl asn1parse -i
    0:d=0  hl=4 l=3594 cons: SEQUENCE
    4:d=1  hl=4 l=3314 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=  17 prim:   INTEGER           :DF5B691B21764A32121C2F378BE596D3
   32:d=2  hl=2 l=  13 cons:   SEQUENCE
   34:d=3  hl=2 l=   9 prim:    OBJECT            :sha256WithRSAEncryption
   45:d=3  hl=2 l=   0 prim:    NULL
   47:d=2  hl=2 l=  59 cons:   SEQUENCE
[...]
   98:d=5  hl=2 l=   3 prim:      OBJECT            :commonName
  103:d=5  hl=2 l=   3 prim:      PRINTABLESTRING   :WR2
  108:d=2  hl=2 l=  30 cons:   SEQUENCE
  110:d=3  hl=2 l=  13 prim:    UTCTIME           :250127083527Z
  125:d=3  hl=2 l=  13 prim:    UTCTIME           :250421083526Z
  140:d=2  hl=2 l=  23 cons:   SEQUENCE
  142:d=3  hl=2 l=  21 cons:    SET
  144:d=4  hl=2 l=  19 cons:     SEQUENCE
  146:d=5  hl=2 l=   3 prim:      OBJECT            :commonName
  151:d=5  hl=2 l=  12 prim:      UTF8STRING        :*.google.com
[...]
```

CVE-2022-0778 & cryptographic data structure editing

March 2022 - CVE-2022-0778 - OpenSSL Advisory

```
Severity
    High
Published
    15 March 2022
Title
    Infinite loop in BN_mod_sqrt() reachable when parsing certificates
Found by
    Tavis Ormandy (Google)
Affected versions
    OpenSSL
        from 3.0.0 before 3.0.2
        from 1.1.1 before 1.1.1n
        from 1.0.2 before 1.0.2zd
```

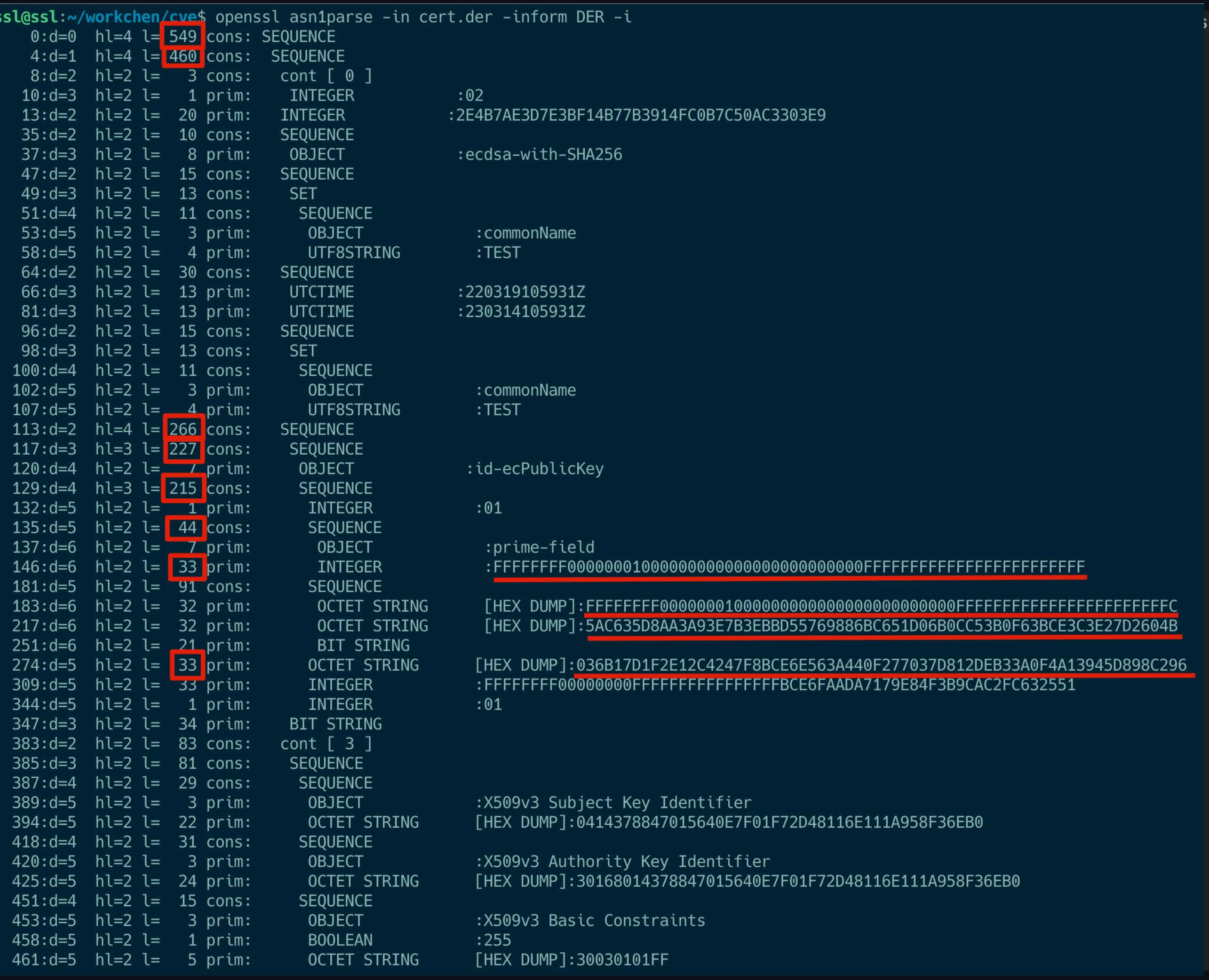
https://openssl-library.org/news/vulnerabilities/index.html#CVE-2022-0778

The first PoC was released:

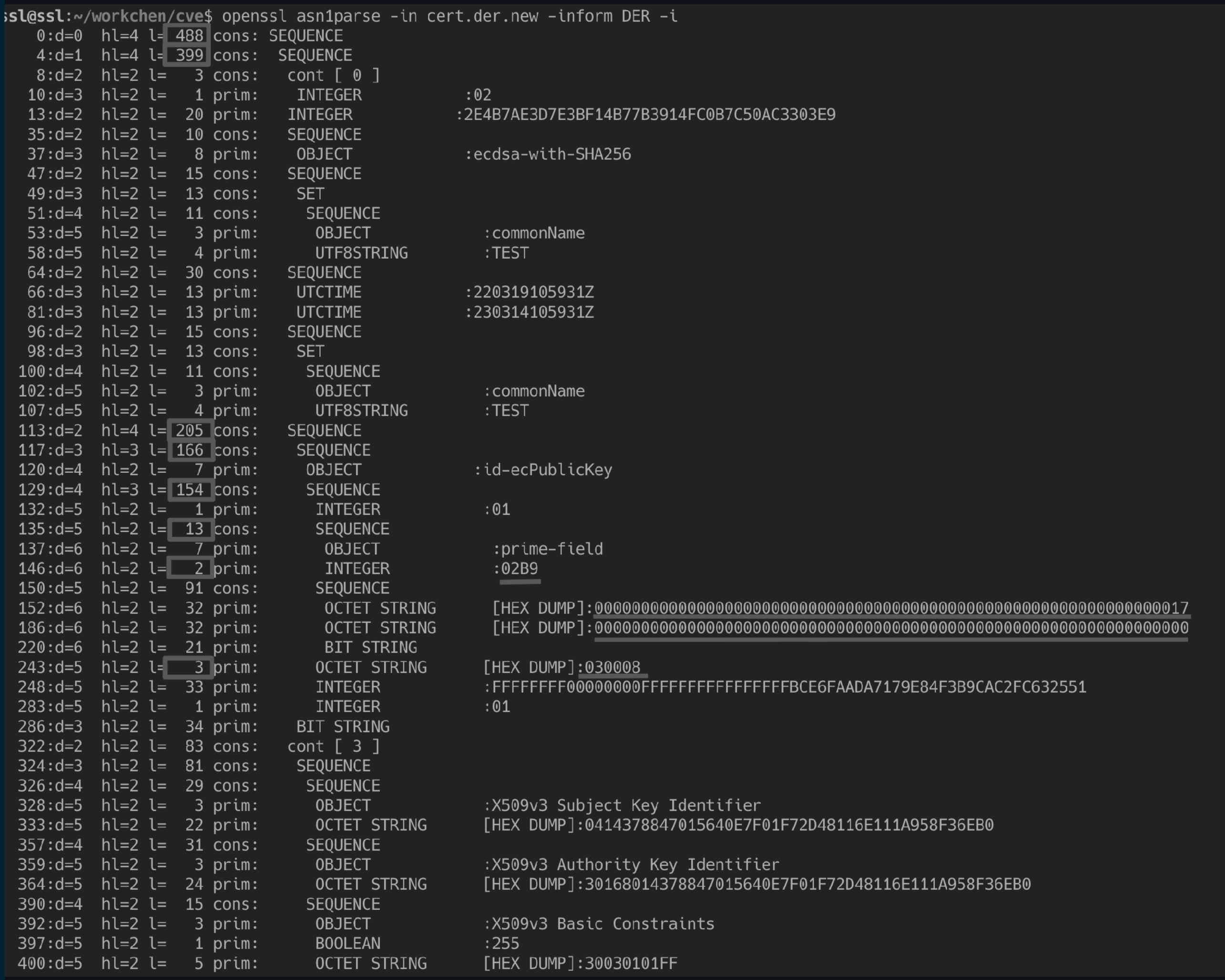
https://github.com/drago-96/CVE-2022-0778

```
The discovered vulnerability triggers an infinite loop in the function
BN_mod_sqrt() of OpenSSL while parsing an elliptic curve key. This means that a
maliciously crafted X.509 certificate can DoS any unpatched server.

The core of the vulnerability is in the parsing of EC keys with points in
compressed format:
[...]
```

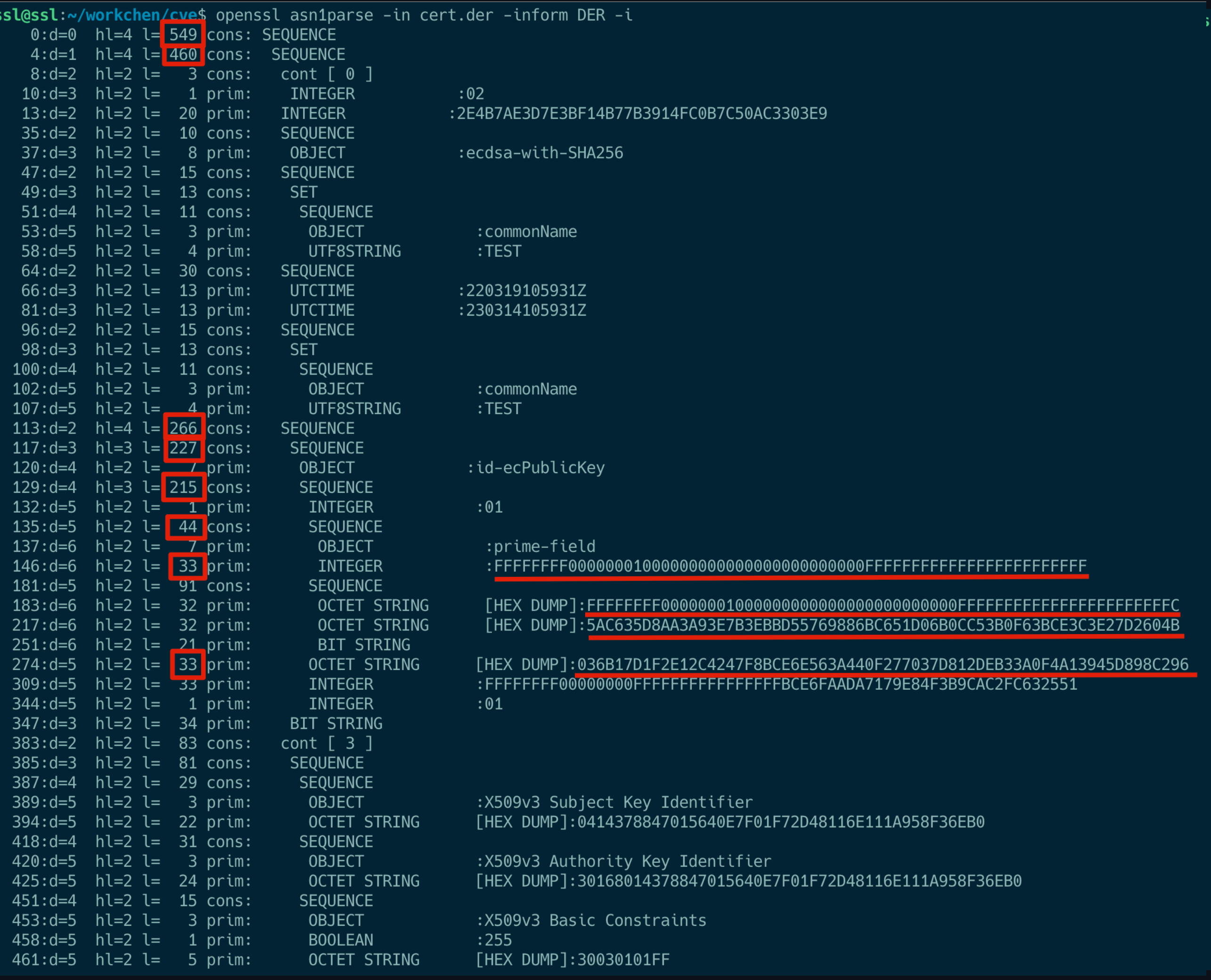
This PoC involves crafting a new X.509 certificate or modifying an existing one



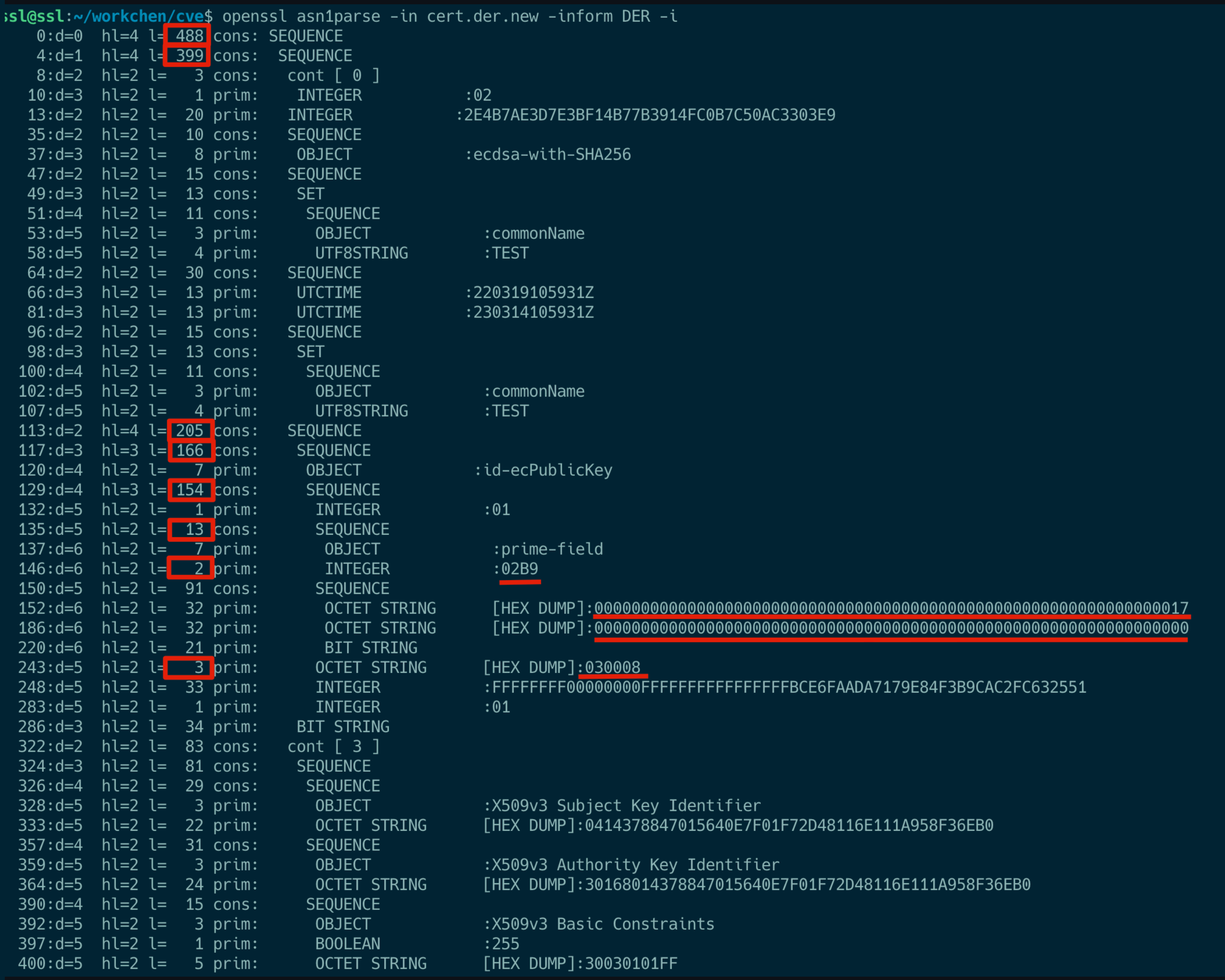
```
ssl@ssl:~/workchen/cve$ openssl asn1parse -in cert.der -inform DER -i
    0:d=0  hl=4 l= 549 cons: SEQUENCE
    4:d=1  hl=4 l= 460 cons:  SEQUENCE
    8:d=2  hl=2 l=   3 cons:   cont [ 0 ]
   10:d=3  hl=2 l=   1 prim:    INTEGER           :02
   13:d=2  hl=2 l=  20 prim:   INTEGER           :2E4B7AE3D7E3BF14B77B3914FC0B7C50AC3303E9
   35:d=2  hl=2 l=  10 cons:   SEQUENCE
   37:d=3  hl=2 l=   8 prim:    OBJECT            :ecdsa-with-SHA256
   47:d=2  hl=2 l=  15 cons:   SEQUENCE
   49:d=3  hl=2 l=  13 cons:    SET
   51:d=4  hl=2 l=  11 cons:     SEQUENCE
   53:d=5  hl=2 l=   3 prim:      OBJECT            :commonName
   58:d=5  hl=2 l=   4 prim:      UTF8STRING        :TEST
   64:d=2  hl=2 l=  30 cons:   SEQUENCE
   66:d=3  hl=2 l=  13 prim:    UTCTIME           :220319105931Z
   81:d=3  hl=2 l=  13 prim:    UTCTIME           :230314105931Z
   96:d=2  hl=2 l=  15 cons:   SEQUENCE
   98:d=3  hl=2 l=  13 cons:    SET
  100:d=4  hl=2 l=  11 cons:     SEQUENCE
  102:d=5  hl=2 l=   3 prim:      OBJECT            :commonName
  107:d=5  hl=2 l=   4 prim:      UTF8STRING        :TEST
  113:d=2  hl=4 l= 266 cons:   SEQUENCE
  117:d=3  hl=3 l= 227 cons:    SEQUENCE
  120:d=4  hl=2 l=   7 prim:     OBJECT            :id-ecPublicKey
  129:d=4  hl=3 l= 215 cons:     SEQUENCE
  132:d=5  hl=2 l=   1 prim:      INTEGER           :01
  135:d=5  hl=2 l=  44 cons:      SEQUENCE
  137:d=6  hl=2 l=   7 prim:       OBJECT            :prime-field
  146:d=6  hl=2 l=  33 prim:       INTEGER           :FFFFFFFF00000001000000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFF
  181:d=5  hl=2 l=  91 cons:      SEQUENCE
  183:d=6  hl=2 l=  32 prim:       OCTET STRING      [HEX DUMP]:FFFFFFFF00000001000000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFC
  217:d=6  hl=2 l=  32 prim:       OCTET STRING      [HEX DUMP]:5AC635D8AA3A93E7B3EBBD55769886BC651D06B0CC53B0F63BCE3C3E27D2604B
  251:d=6  hl=2 l=  21 prim:       BIT STRING
  274:d=5  hl=2 l=  33 prim:      OCTET STRING      [HEX DUMP]:036B17D1F2E12C4247F8BCE6E563A440F277037D812DEB33A0F4A13945D898C296
  309:d=5  hl=2 l=  33 prim:      INTEGER           :FFFFFFFF00000000FFFFFFFFFFFFFFFFBCE6FAADA7179E84F3B9CAC2FC632551
  344:d=5  hl=2 l=   1 prim:      INTEGER           :01
  347:d=3  hl=2 l=  34 prim:    BIT STRING
  383:d=2  hl=2 l=  83 cons:   cont [ 3 ]
  385:d=3  hl=2 l=  81 cons:    SEQUENCE
  387:d=4  hl=2 l=  29 cons:     SEQUENCE
  389:d=5  hl=2 l=   3 prim:      OBJECT            :X509v3 Subject Key Identifier
  394:d=5  hl=2 l=  22 prim:      OCTET STRING      [HEX DUMP]:0414378847015640E7F01F72D48116E111A958F36EB0
  418:d=4  hl=2 l=  31 cons:     SEQUENCE
  420:d=5  hl=2 l=   3 prim:      OBJECT            :X509v3 Authority Key Identifier
  425:d=5  hl=2 l=  24 prim:      OCTET STRING      [HEX DUMP]:30168014378847015640E7F01F72D48116E111A958F36EB0
  451:d=4  hl=2 l=  15 cons:     SEQUENCE
  453:d=5  hl=2 l=   3 prim:      OBJECT            :X509v3 Basic Constraints
  458:d=5  hl=2 l=   1 prim:      BOOLEAN           :255
  461:d=5  hl=2 l=   5 prim:      OCTET STRING      [HEX DUMP]:30030101FF
```



```
ssl@ssl:~/workchen/cve$ openssl asn1parse -in cert.der.new -inform DER -i
    0:d=0  hl=4 l= 488 cons: SEQUENCE
    4:d=1  hl=4 l= 399 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=  20 prim:   INTEGER           :2E4B7AE3D7E3BF14B77B3914FC0B7C50AC3303E9
   35:d=2  hl=2 l=  10 cons:   SEQUENCE
   37:d=3  hl=2 l=   8 prim:    OBJECT            :ecdsa-with-SHA256
   47:d=2  hl=2 l=  15 cons:   SEQUENCE
   49:d=3  hl=2 l=  13 cons:    SET
   51:d=4  hl=2 l=  11 cons:     SEQUENCE
   53:d=5  hl=2 l=   3 prim:      OBJECT            :commonName
   58:d=5  hl=2 l=   4 prim:      UTF8STRING        :TEST
   64:d=2  hl=2 l=  30 cons:   SEQUENCE
   66:d=3  hl=2 l=  13 prim:    UTCTIME           :220319105931Z
   81:d=3  hl=2 l=  13 prim:    UTCTIME           :230314105931Z
   96:d=2  hl=2 l=  15 cons:   SEQUENCE
   98:d=3  hl=2 l=  13 cons:    SET
  100:d=4  hl=2 l=  11 cons:     SEQUENCE
  102:d=5  hl=2 l=   3 prim:      OBJECT            :commonName
  107:d=5  hl=2 l=   4 prim:      UTF8STRING        :TEST
  113:d=2  hl=4 l= 205 cons:   SEQUENCE
  117:d=3  hl=3 l= 166 cons:    SEQUENCE
  120:d=4  hl=2 l=   7 prim:     OBJECT            :id-ecPublicKey
  129:d=4  hl=3 l= 154 cons:     SEQUENCE
  132:d=5  hl=2 l=   1 prim:      INTEGER           :01
  135:d=5  hl=2 l=  13 cons:      SEQUENCE
  137:d=6  hl=2 l=   7 prim:       OBJECT            :prime-field
  146:d=6  hl=2 l=   2 prim:       INTEGER           :02B9
  150:d=5  hl=2 l=  91 cons:      SEQUENCE
  152:d=6  hl=2 l=  32 prim:       OCTET STRING      [HEX DUMP]:0000000000000000000000000000000000000000000000000000000000000017
  186:d=6  hl=2 l=  32 prim:       OCTET STRING      [HEX DUMP]:0000000000000000000000000000000000000000000000000000000000000000
  220:d=6  hl=2 l=  21 prim:       BIT STRING
  243:d=5  hl=2 l=   3 prim:      OCTET STRING      [HEX DUMP]:030008
  248:d=5  hl=2 l=  33 prim:      INTEGER           :FFFFFFFF00000000FFFFFFFFFFFFFFFFBCE6FAADA7179E84F3B9CAC2FC632551
  283:d=5  hl=2 l=   1 prim:      INTEGER           :01
  286:d=3  hl=2 l=  34 prim:    BIT STRING
  322:d=2  hl=2 l=  83 cons:   cont [ 3 ]
  324:d=3  hl=2 l=  81 cons:    SEQUENCE
  326:d=4  hl=2 l=  29 cons:     SEQUENCE
  328:d=5  hl=2 l=   3 prim:      OBJECT            :X509v3 Subject Key Identifier
  333:d=5  hl=2 l=  22 prim:      OCTET STRING      [HEX DUMP]:0414378847015640E7F01F72D48116E111A958F36EB0
  357:d=4  hl=2 l=  31 cons:     SEQUENCE
  359:d=5  hl=2 l=   3 prim:      OBJECT            :X509v3 Authority Key Identifier
  364:d=5  hl=2 l=  24 prim:      OCTET STRING      [HEX DUMP]:30168014378847015640E7F01F72D48116E111A958F36EB0
  390:d=4  hl=2 l=  15 cons:     SEQUENCE
  392:d=5  hl=2 l=   3 prim:      OBJECT            :X509v3 Basic Constraints
  397:d=5  hl=2 l=   1 prim:      BOOLEAN           :255
  400:d=5  hl=2 l=   5 prim:      OCTET STRING      [HEX DUMP]:30030101FF
```

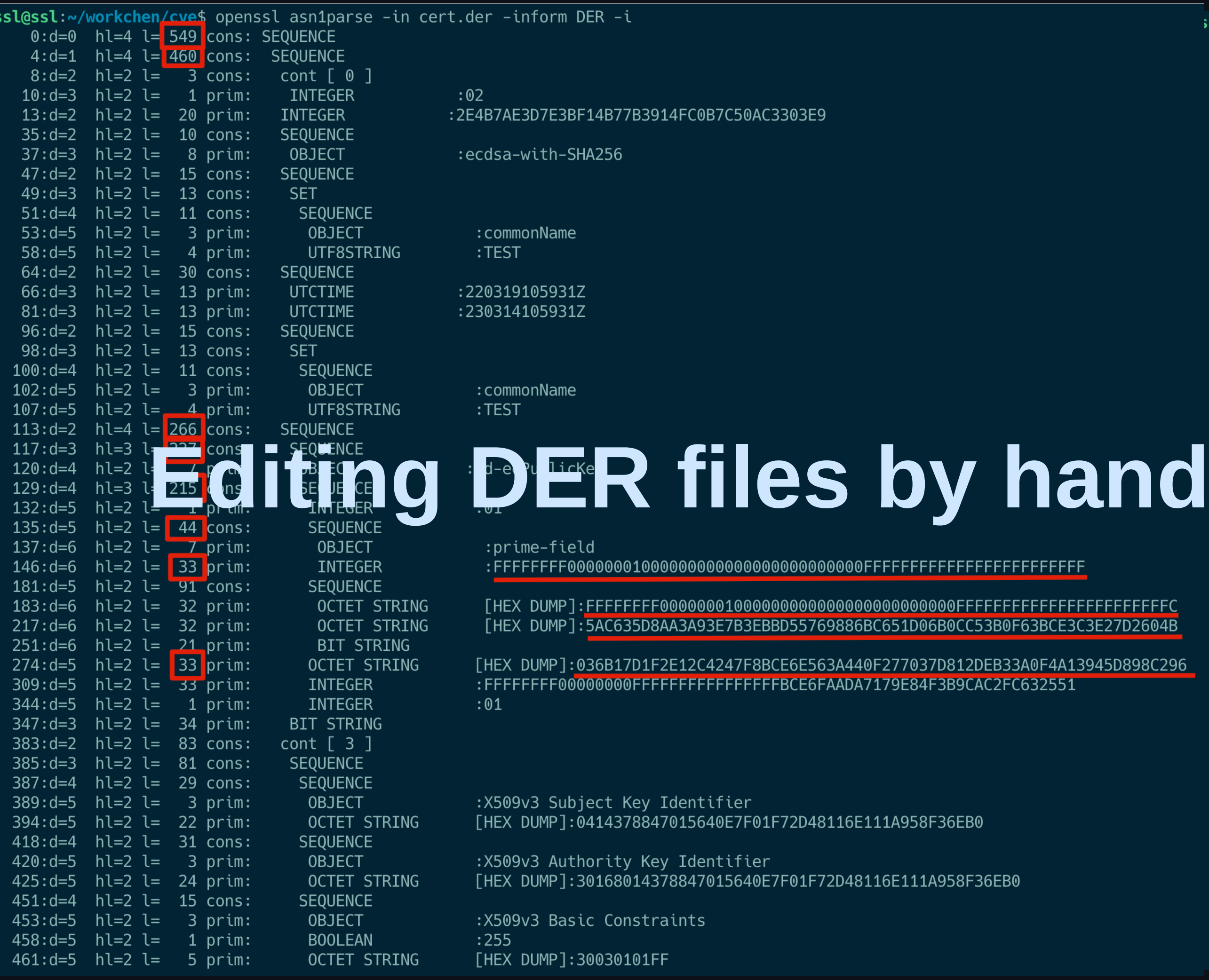


```
sl@ssl:~/workchen/cve$ openssl asn1parse -in cert.der -inform DER -i
    0:d=0  hl=4 l= 549 cons: SEQUENCE
    4:d=1  hl=4 l= 460 cons:  SEQUENCE
    8:d=2  hl=2 l=   3 cons:   cont [ 0 ]
   10:d=3  hl=2 l=   1 prim:    INTEGER           :02
   13:d=2  hl=2 l=  20 prim:   INTEGER           :2E4B7AE3D7E3BF14B77B3914FC0B7C50AC3303E9
   35:d=2  hl=2 l=  10 cons:   SEQUENCE
   37:d=3  hl=2 l=   8 prim:    OBJECT            :ecdsa-with-SHA256
   47:d=2  hl=2 l=  15 cons:   SEQUENCE
   49:d=3  hl=2 l=  13 cons:    SET
   51:d=4  hl=2 l=  11 cons:     SEQUENCE
   53:d=5  hl=2 l=   3 prim:      OBJECT            :commonName
   58:d=5  hl=2 l=   4 prim:      UTF8STRING        :TEST
   64:d=2  hl=2 l=  30 cons:   SEQUENCE
   66:d=3  hl=2 l=  13 prim:    UTCTIME           :220319105931Z
   81:d=3  hl=2 l=  13 prim:    UTCTIME           :230314105931Z
   96:d=2  hl=2 l=  15 cons:   SEQUENCE
   98:d=3  hl=2 l=  13 cons:    SET
  100:d=4  hl=2 l=  11 cons:     SEQUENCE
  102:d=5  hl=2 l=   3 prim:      OBJECT            :commonName
  107:d=5  hl=2 l=   4 prim:      UTF8STRING        :TEST
  113:d=2  hl=4 l= 266 cons:   SEQUENCE
  117:d=3  hl=3 l= 227 cons:    SEQUENCE
  120:d=4  hl=2 l=   7 prim:     OBJECT            :id-ecPublicKey
  129:d=4  hl=3 l= 215 cons:     SEQUENCE
  132:d=5  hl=2 l=   1 prim:      INTEGER           :01
  135:d=5  hl=2 l=  44 cons:      SEQUENCE
  137:d=6  hl=2 l=   7 prim:       OBJECT            :prime-field
  146:d=6  hl=2 l=  33 prim:       INTEGER           :FFFFFFFF00000001000000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFF
  181:d=5  hl=2 l=  91 cons:      SEQUENCE
  183:d=6  hl=2 l=  32 prim:       OCTET STRING      [HEX DUMP]:FFFFFFFF00000001000000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFC
  217:d=6  hl=2 l=  32 prim:       OCTET STRING      [HEX DUMP]:5AC635D8AA3A93E7B3EBBD55769886BC651D06B0CC53B0F63BCE3C3E27D2604B
  251:d=6  hl=2 l=  21 prim:       BIT STRING
  274:d=5  hl=2 l=  33 prim:      OCTET STRING      [HEX DUMP]:036B17D1F2E12C4247F8BCE6E563A440F277037D812DEB33A0F4A13945D898C296
  309:d=5  hl=2 l=  33 prim:      INTEGER           :FFFFFFFF00000000FFFFFFFFFFFFFFFFBCE6FAADA7179E84F3B9CAC2FC632551
  344:d=5  hl=2 l=   1 prim:      INTEGER           :01
  347:d=3  hl=2 l=  34 prim:    BIT STRING
  383:d=2  hl=2 l=  83 cons:   cont [ 3 ]
  385:d=3  hl=2 l=  81 cons:    SEQUENCE
  387:d=4  hl=2 l=  29 cons:     SEQUENCE
  389:d=5  hl=2 l=   3 prim:      OBJECT            :X509v3 Subject Key Identifier
  394:d=5  hl=2 l=  22 prim:      OCTET STRING      [HEX DUMP]:0414378847015640E7F01F72D48116E111A958F36EB0
  418:d=4  hl=2 l=  31 cons:     SEQUENCE
  420:d=5  hl=2 l=   3 prim:      OBJECT            :X509v3 Authority Key Identifier
  425:d=5  hl=2 l=  24 prim:      OCTET STRING      [HEX DUMP]:30168014378847015640E7F01F72D48116E111A958F36EB0
  451:d=4  hl=2 l=  15 cons:     SEQUENCE
  453:d=5  hl=2 l=   3 prim:      OBJECT            :X509v3 Basic Constraints
  458:d=5  hl=2 l=   1 prim:      BOOLEAN           :255
  461:d=5  hl=2 l=   5 prim:      OCTET STRING      [HEX DUMP]:30030101FF
```

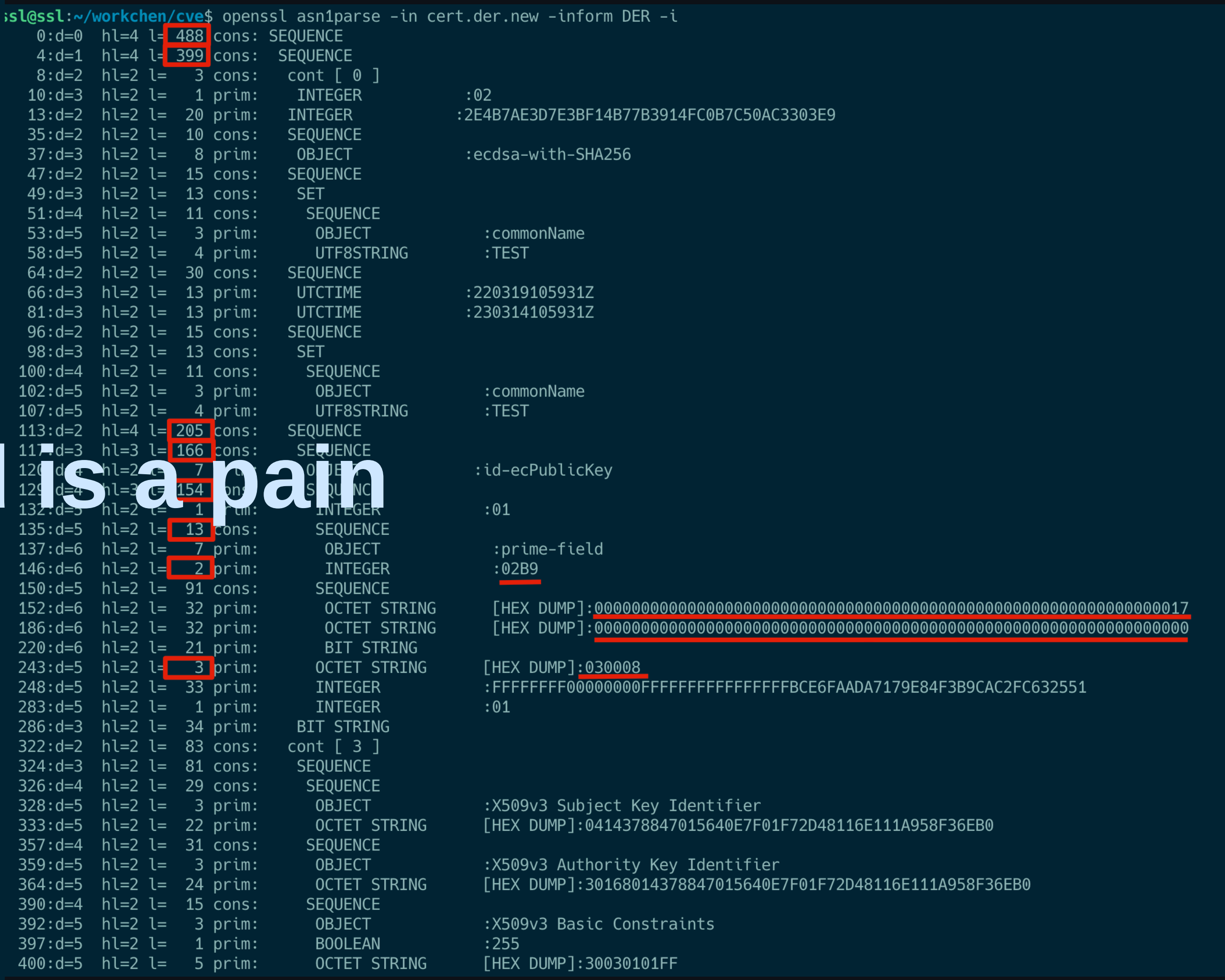


```
sl@ssl:~/workchen/cve$ openssl asn1parse -in cert.der.new -inform DER -i
    0:d=0  hl=4 l= 488 cons: SEQUENCE
    4:d=1  hl=4 l= 399 cons:  SEQUENCE
    8:d=2  hl=2 l=   3 cons:   cont [ 0 ]
   10:d=3  hl=2 l=   1 prim:    INTEGER           :02
   13:d=2  hl=2 l=  20 prim:   INTEGER           :2E4B7AE3D7E3BF14B77B3914FC0B7C50AC3303E9
   35:d=2  hl=2 l=  10 cons:   SEQUENCE
   37:d=3  hl=2 l=   8 prim:    OBJECT            :ecdsa-with-SHA256
   47:d=2  hl=2 l=  15 cons:   SEQUENCE
   49:d=3  hl=2 l=  13 cons:    SET
   51:d=4  hl=2 l=  11 cons:     SEQUENCE
   53:d=5  hl=2 l=   3 prim:      OBJECT            :commonName
   58:d=5  hl=2 l=   4 prim:      UTF8STRING        :TEST
   64:d=2  hl=2 l=  30 cons:   SEQUENCE
   66:d=3  hl=2 l=  13 prim:    UTCTIME           :220319105931Z
   81:d=3  hl=2 l=  13 prim:    UTCTIME           :230314105931Z
   96:d=2  hl=2 l=  15 cons:   SEQUENCE
   98:d=3  hl=2 l=  13 cons:    SET
  100:d=4  hl=2 l=  11 cons:     SEQUENCE
  102:d=5  hl=2 l=   3 prim:      OBJECT            :commonName
  107:d=5  hl=2 l=   4 prim:      UTF8STRING        :TEST
  113:d=2  hl=4 l= 205 cons:   SEQUENCE
  117:d=3  hl=3 l= 166 cons:    SEQUENCE
  120:d=4  hl=2 l=   7 prim:     OBJECT            :id-ecPublicKey
  129:d=4  hl=3 l= 154 cons:     SEQUENCE
  132:d=5  hl=2 l=   1 prim:      INTEGER           :01
  135:d=5  hl=2 l=  13 cons:      SEQUENCE
  137:d=6  hl=2 l=   7 prim:       OBJECT            :prime-field
  146:d=6  hl=2 l=   2 prim:       INTEGER           :02B9
  150:d=5  hl=2 l=  91 cons:      SEQUENCE
  152:d=6  hl=2 l=  32 prim:       OCTET STRING      [HEX DUMP]:0000000000000000000000000000000000000000000000000000000000000017
  186:d=6  hl=2 l=  32 prim:       OCTET STRING      [HEX DUMP]:0000000000000000000000000000000000000000000000000000000000000000
  220:d=6  hl=2 l=  21 prim:       BIT STRING
  243:d=5  hl=2 l=   3 prim:      OCTET STRING      [HEX DUMP]:030008
  248:d=5  hl=2 l=  33 prim:      INTEGER           :FFFFFFFF00000000FFFFFFFFFFFFFFFFBCE6FAADA7179E84F3B9CAC2FC632551
  283:d=5  hl=2 l=   1 prim:      INTEGER           :01
  286:d=3  hl=2 l=  34 prim:    BIT STRING
  322:d=2  hl=2 l=  83 cons:   cont [ 3 ]
  324:d=3  hl=2 l=  81 cons:    SEQUENCE
  326:d=4  hl=2 l=  29 cons:     SEQUENCE
  328:d=5  hl=2 l=   3 prim:      OBJECT            :X509v3 Subject Key Identifier
  333:d=5  hl=2 l=  22 prim:      OCTET STRING      [HEX DUMP]:0414378847015640E7F01F72D48116E111A958F36EB0
  357:d=4  hl=2 l=  31 cons:     SEQUENCE
  359:d=5  hl=2 l=   3 prim:      OBJECT            :X509v3 Authority Key Identifier
  364:d=5  hl=2 l=  24 prim:      OCTET STRING      [HEX DUMP]:30168014378847015640E7F01F72D48116E111A958F36EB0
  390:d=4  hl=2 l=  15 cons:     SEQUENCE
  392:d=5  hl=2 l=   3 prim:      OBJECT            :X509v3 Basic Constraints
  397:d=5  hl=2 l=   1 prim:      BOOLEAN           :255
  400:d=5  hl=2 l=   5 prim:      OCTET STRING      [HEX DUMP]:30030101FF
```

Editing DER files by hand is a pain



```
sl@ssl:~/workchen/cve$ openssl asn1parse -in cert.der -inform DER -i
    0:d=0  hl=4 l= 549 cons: SEQUENCE
    4:d=1  hl=4 l= 460 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=  20 prim:   INTEGER           :2E4B7AE3D7E3BF14B77B3914FC0B7C50AC3303E9
   35:d=2  hl=2 l=  10 cons:   SEQUENCE
   37:d=3  hl=2 l=   8 prim:    OBJECT            :ecdsa-with-SHA256
   47:d=2  hl=2 l=  15 cons:   SEQUENCE
   49:d=3  hl=2 l=  13 cons:    SET
   51:d=4  hl=2 l=  11 cons:     SEQUENCE
   53:d=5  hl=2 l=   3 prim:      OBJECT            :commonName
   58:d=5  hl=2 l=   4 prim:      UTF8STRING        :TEST
   64:d=2  hl=2 l=  30 cons:   SEQUENCE
   66:d=3  hl=2 l=  13 prim:    UTCTIME           :220319105931Z
   81:d=3  hl=2 l=  13 prim:    UTCTIME           :230314105931Z
   96:d=2  hl=2 l=  15 cons:   SEQUENCE
   98:d=3  hl=2 l=  13 cons:    SET
  100:d=4  hl=2 l=  11 cons:     SEQUENCE
  102:d=5  hl=2 l=   3 prim:      OBJECT            :commonName
  107:d=5  hl=2 l=   4 prim:      UTF8STRING        :TEST
  113:d=2  hl=4 l= 266 cons:   SEQUENCE
  117:d=3  hl=3 l= 227 cons:    SEQUENCE
  120:d=4  hl=2 l=   7 prim:     OBJECT            :id-ecPublicKey
  129:d=4  hl=3 l= 215 cons:     SEQUENCE
  132:d=5  hl=2 l=   1 prim:      INTEGER           :01
  135:d=5  hl=2 l=  44 cons:      SEQUENCE
  137:d=6  hl=2 l=   7 prim:       OBJECT            :prime-field
  146:d=6  hl=2 l=  33 prim:       INTEGER           :FFFFFFFF00000001000000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFF
  181:d=5  hl=2 l=  91 cons:      SEQUENCE
  183:d=6  hl=2 l=  32 prim:       OCTET STRING      [HEX DUMP]:FFFFFFFF00000001000000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFC
  217:d=6  hl=2 l=  32 prim:       OCTET STRING      [HEX DUMP]:5AC635D8AA3A93E7B3EBBD55769886BC651D06B0CC53B0F63BCE3C3E27D2604B
  251:d=6  hl=2 l=  21 prim:       BIT STRING
  274:d=5  hl=2 l=  33 prim:      OCTET STRING      [HEX DUMP]:036B17D1F2E12C4247F8BCE6E563A440F277037D812DEB33A0F4A13945D898C296
  309:d=5  hl=2 l=  33 prim:      INTEGER           :FFFFFFFF00000000FFFFFFFFFFFFFFFFBCE6FAADA7179E84F3B9CAC2FC632551
  344:d=5  hl=2 l=   1 prim:      INTEGER           :01
  347:d=3  hl=2 l=  34 prim:    BIT STRING
  383:d=2  hl=2 l=  83 cons:   cont [ 3 ]
  385:d=3  hl=2 l=  81 cons:    SEQUENCE
  387:d=4  hl=2 l=  29 cons:     SEQUENCE
  389:d=5  hl=2 l=   3 prim:      OBJECT            :X509v3 Subject Key Identifier
  394:d=5  hl=2 l=  22 prim:      OCTET STRING      [HEX DUMP]:0414378847015640E7F01F72D48116E111A958F36EB0
  418:d=4  hl=2 l=  31 cons:     SEQUENCE
  420:d=5  hl=2 l=   3 prim:      OBJECT            :X509v3 Authority Key Identifier
  425:d=5  hl=2 l=  24 prim:      OCTET STRING      [HEX DUMP]:30168014378847015640E7F01F72D48116E111A958F36EB0
  451:d=4  hl=2 l=  15 cons:     SEQUENCE
  453:d=5  hl=2 l=   3 prim:      OBJECT            :X509v3 Basic Constraints
  458:d=5  hl=2 l=   1 prim:      BOOLEAN           :255
  461:d=5  hl=2 l=   5 prim:      OCTET STRING      [HEX DUMP]:30030101FF
```



```
sl@ssl:~/workchen/cve$ openssl asn1parse -in cert.der.new -inform DER -i
    0:d=0  hl=4 l= 488 cons: SEQUENCE
    4:d=1  hl=4 l= 399 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=  20 prim:   INTEGER           :2E4B7AE3D7E3BF14B77B3914FC0B7C50AC3303E9
   35:d=2  hl=2 l=  10 cons:   SEQUENCE
   37:d=3  hl=2 l=   8 prim:    OBJECT            :ecdsa-with-SHA256
   47:d=2  hl=2 l=  15 cons:   SEQUENCE
   49:d=3  hl=2 l=  13 cons:    SET
   51:d=4  hl=2 l=  11 cons:     SEQUENCE
   53:d=5  hl=2 l=   3 prim:      OBJECT            :commonName
   58:d=5  hl=2 l=   4 prim:      UTF8STRING        :TEST
   64:d=2  hl=2 l=  30 cons:   SEQUENCE
   66:d=3  hl=2 l=  13 prim:    UTCTIME           :220319105931Z
   81:d=3  hl=2 l=  13 prim:    UTCTIME           :230314105931Z
   96:d=2  hl=2 l=  15 cons:   SEQUENCE
   98:d=3  hl=2 l=  13 cons:    SET
  100:d=4  hl=2 l=  11 cons:     SEQUENCE
  102:d=5  hl=2 l=   3 prim:      OBJECT            :commonName
  107:d=5  hl=2 l=   4 prim:      UTF8STRING        :TEST
  113:d=2  hl=4 l= 205 cons:   SEQUENCE
  117:d=3  hl=3 l= 166 cons:    SEQUENCE
  120:d=4  hl=2 l=   7 prim:     OBJECT            :id-ecPublicKey
  129:d=4  hl=3 l= 154 cons:     SEQUENCE
  132:d=5  hl=2 l=   1 prim:      INTEGER           :01
  135:d=5  hl=2 l=  13 cons:      SEQUENCE
  137:d=6  hl=2 l=   7 prim:       OBJECT            :prime-field
  146:d=6  hl=2 l=   2 prim:       INTEGER           :02B9
  150:d=5  hl=2 l=  91 cons:      SEQUENCE
  152:d=6  hl=2 l=  32 prim:       OCTET STRING      [HEX DUMP]:0000000000000000000000000000000000000000000000000000000000000017
  186:d=6  hl=2 l=  32 prim:       OCTET STRING      [HEX DUMP]:0000000000000000000000000000000000000000000000000000000000000000
  220:d=6  hl=2 l=  21 prim:       BIT STRING
  243:d=5  hl=2 l=   3 prim:      OCTET STRING      [HEX DUMP]:030008
  248:d=5  hl=2 l=  33 prim:      INTEGER           :FFFFFFFF00000000FFFFFFFFFFFFFFFFBCE6FAADA7179E84F3B9CAC2FC632551
  283:d=5  hl=2 l=   1 prim:      INTEGER           :01
  286:d=3  hl=2 l=  34 prim:    BIT STRING
  322:d=2  hl=2 l=  83 cons:   cont [ 3 ]
  324:d=3  hl=2 l=  81 cons:    SEQUENCE
  326:d=4  hl=2 l=  29 cons:     SEQUENCE
  328:d=5  hl=2 l=   3 prim:      OBJECT            :X509v3 Subject Key Identifier
  333:d=5  hl=2 l=  22 prim:      OCTET STRING      [HEX DUMP]:0414378847015640E7F01F72D48116E111A958F36EB0
  357:d=4  hl=2 l=  31 cons:     SEQUENCE
  359:d=5  hl=2 l=   3 prim:      OBJECT            :X509v3 Authority Key Identifier
  364:d=5  hl=2 l=  24 prim:      OCTET STRING      [HEX DUMP]:30168014378847015640E7F01F72D48116E111A958F36EB0
  390:d=4  hl=2 l=  15 cons:     SEQUENCE
  392:d=5  hl=2 l=   3 prim:      OBJECT            :X509v3 Basic Constraints
  397:d=5  hl=2 l=   1 prim:      BOOLEAN           :255
  400:d=5  hl=2 l=   5 prim:      OCTET STRING      [HEX DUMP]:30030101FF
```

asn1template.pl - my solution to this problem

```
$ man openssl-asn1parse
[...]
-genstr string, -genconf file
    Generate encoded data based on string, file or both
    using ASN1_generate_nconf(3) format. [...]
[...]
```

Presented at PTS 2023 and Hack.lu 2023 (as a Lightning Talk)

https://github.com/wllm-rbnt/asn1template

asn1template.pl - how it works

Internal structure of the script:

- Reads the output of the asn1parse OpenSSL app
  (a representation of the tree structure)
- Reconstructs the ASN.1 structure tree
- Traverses the tree recursively, depth-first
- and writes 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 only

openssl asn1parse command & asn1template.pl

```
$ 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
```

```
$ asn1template.pl test.der | tee test.tpl
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
```

Using openssl asn1parse -genconf

```
$ openssl asn1parse -genconf test.tpl -out test_reconstructed.der
```

```
$ openssl asn1parse -in test_reconstructed.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
```

```
$ diff test.der test_reconstructed.der
$ echo $?
0
```

Editing the template

```
$ sed -i -e 's/INTEGER:0x01234567/PRINTABLESTRING:HELLO/' test.tpl
$ openssl asn1parse -genconf test.tpl -out test_reconstructed_2.der -i
 0:d=0  hl=2 l=  19 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=   7 cons:  SEQUENCE
14:d=2  hl=2 l=   5 prim:   PRINTABLESTRING   :HELLO
```

```
$ openssl asn1parse -in test_reconstructed.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
```

The story behind this talk

Hack.lu 2024 ...

Would you present a lightning talk ?

Idea ! 💡

I remembered writing this in the README file of asn1template's repository:

```
"Line feeds in OCTET STRINGs break the conversion."
```

This happens because asn1template treats each line of asn1parse as a fully contained item, line feeds being considered as the item separator.

⇨ It's not always the case

Idea ! 💡

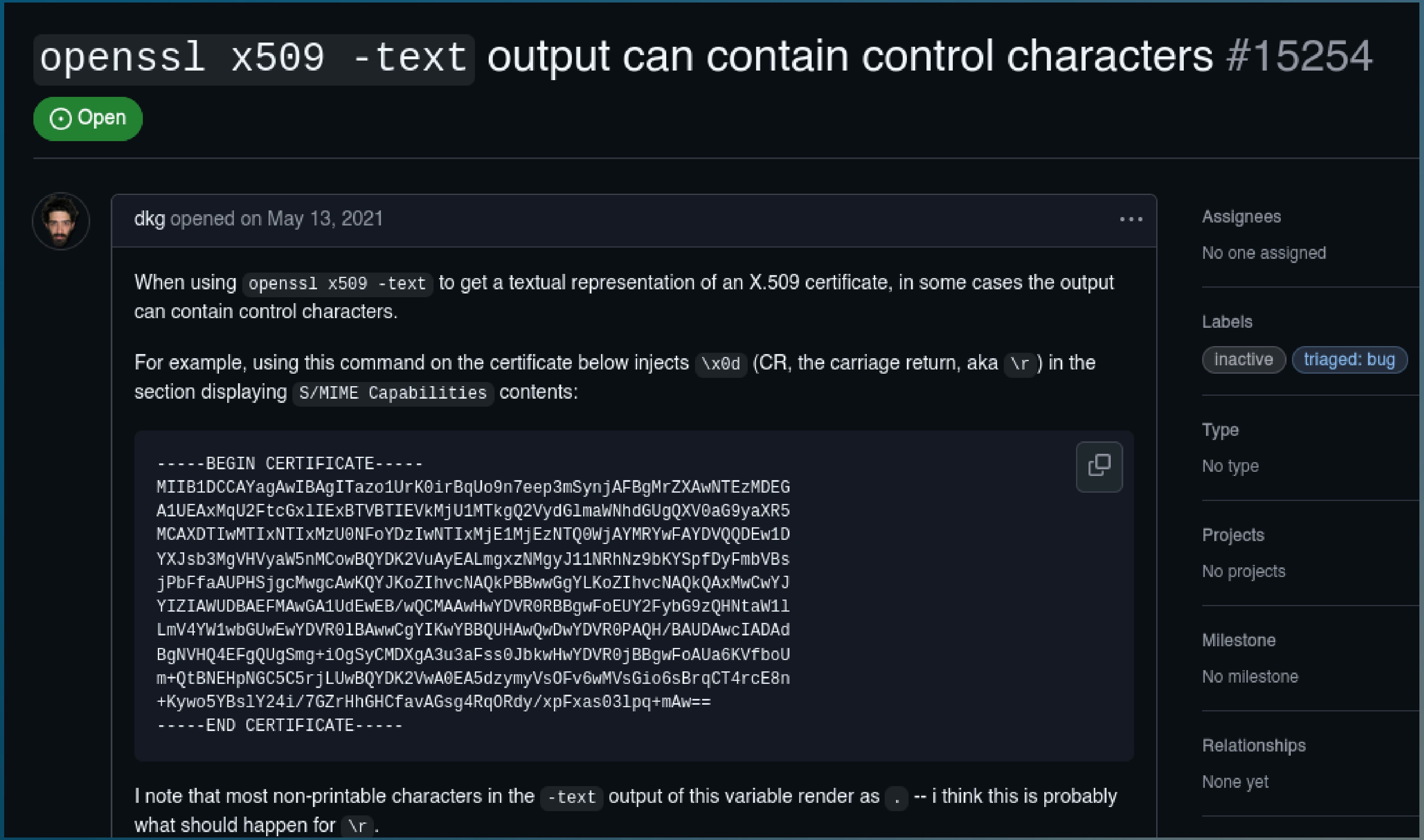
Corpus of 1.7M X.509 Certificates:

https://github.com/johndoe31415/x509-cert-testcorpus

Example of broken structure:

```
[...]
  651:d=4  hl=2 l=  15 cons: SEQUENCE
  653:d=5  hl=2 l=   3 prim:  OBJECT            :X509v3 Certificate Policies
  658:d=5  hl=2 l=   8 prim:  OCTET STRING      :AAAA
BBBB
  668:d=4  hl=2 l=  76 cons: SEQUENCE
  670:d=5  hl=2 l=   3 prim:  OBJECT            :X509v3 CRL Distribution Points
  675:d=5  hl=2 l=  69 prim:  OCTET STRING      [HEX DUMP]:30433041A03FA03D863B68747470[...]
[...]
```

Similar bug reported in 2021



`openssl x509 -text` output can contain control characters #15254

Open

dkg opened on May 13, 2021

When using `openssl x509 -text` to get a textual representation of an X.509 certificate, in some cases the output can contain control characters.

For example, using this command on the certificate below injects `\x0d` (CR, the carriage return, aka `\r`) in the section displaying `S/MIME Capabilities` contents:

```
-----BEGIN CERTIFICATE-----
MIIB1DCCAYagAwIBAgITazo1UrK0irBqUo9n7eep3mSynjAFBgMrZXAwNTEzMDEG
A1UEAxMqU2FtcGxlIExBTVBTIEVkMjU1MTkgQ2VydGlmaWNhdGUgQXV0aG9yaXR5
MCAXDTIwMTIxNTIxMzU0NFoYDzIwNTIxMjE1MjEzNTQ0WjAYMRYwFAYDVQQDEw1D
YXJsb3MgVHVyaW5nMCowBQYDK2VuAyEALmgxzNMgyJ11NRhNz9bKYSpfDyFmbVBs
jPbFfaAUPHSjgcMwgcAwKQYJKoZIhvcNAQkPBBwwGgYLKoZIhvcNAQkQAxMwCwYJ
YIZIAWUDBAEFMAwGA1UdEwEB/wQCMAAwHwYDVR0RBBgwFoEUY2FybG9zQHNtaW1l
LmV4YW1wbGUwEwYDVR01BAwwCgYIKwYBBQUHAwQwDwYDVR0PAQH/BAUDAwcIADAd
BgNVHQ4EFgQUgSmg+iOgSyCMDXgA3u3aFss0JbkwHwYDVR0jBBgwFoAUa6KVfboU
m+QtBNEHpNGC5C5rjLUwBQYDK2VwA0EA5dzymyVsOFv6wMVsGio6sBrqCT4rcE8n
+Kywo5YBslY24i/7GZrHhGHCfavAGsg4RqORdy/xpFxas03lpq+mAw==
-----END CERTIFICATE-----
```

I note that most non-printable characters in the `-text` output of this variable render as `.` -- i think this is probably what should happen for `\r`.

Assignees
No one assigned

Labels
inactive triaged: bug

Type
No type

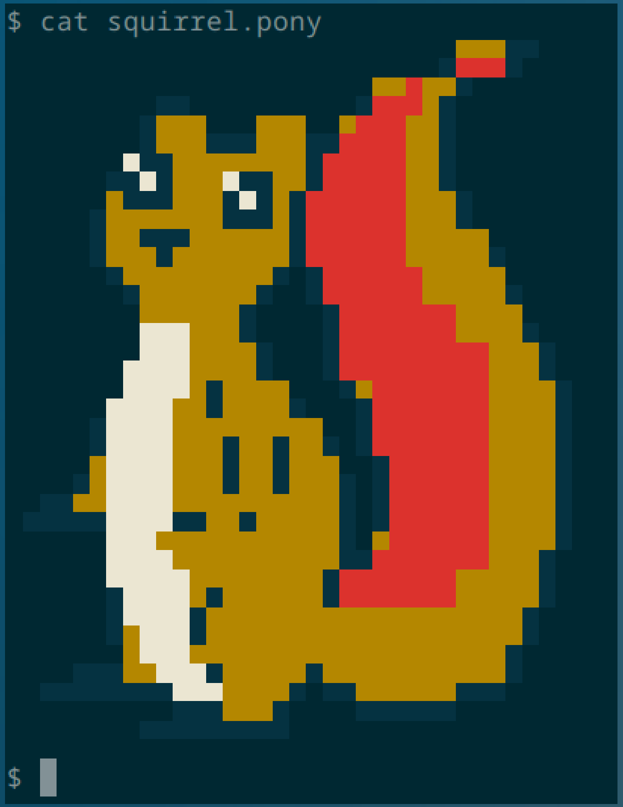
Projects
No projects

Milestone
No milestone

Relationships
None yet

Experiments & findings

Goal: embed some ANSI art into text fields of a certificate



```
$ cat squirrel.pony
$
```

Requirement for this to work:

terminal escape sequences, control and Unicode characters must be printed raw

2 approaches:

- use a CSR configuration file (cfr. man openssl-req)
- generate a binary structure from a template with a placeholder of same length as implanted data, then replace in binary file directly (adjust signature)

ANSI art source: https://github.com/erkin/ponysay

Terminal Control Characters & Sequences

- Control Characters (cfr. man ascii)
- Control Sequence Introducer (CSI) ⇨ introduced by ^[
- Operating System Command (OSC) ⇨ introduced by ^]

References:

- https://en.wikipedia.org/wiki/ANSI_escape_code
- "Weaponizing Plain Text ANSI Escape Sequences as a Forensic Nightmare" by STÖK ⇨ https://www.youtube.com/watch?v=KD1jEIMSjA4

Control Characters

Control characters are the ones with hex value less than 0x20 (cfr. man ascii)

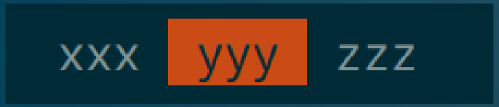
```
Hex   Char                          Ctrl-Key
────────────────────────────────────────────
00    NUL '\0' (null character)     ^@
[...]
07    BEL '\a' (bell)               ^G
08    BS  '\b' (backspace)          ^H
09    HT  '\t' (horizontal tab)     ^I
0A    LF  '\n' (new line)           ^J
0B    VT  '\v' (vertical tab)       ^K
0C    FF  '\f' (form feed)          ^L
0D    CR  '\r' (carriage ret)       ^M
[...]
1B    ESC (escape)                  ^[
[...]
20    SPACE                        N/A
```

Mainly used for cursor movement, but also the entry point for Terminal Escape Sequences

Terminal Escape Sequences

Terminal Escape Sequences are used for color changes, cursor movements, terminal configuration, ...

```
            /---------------------------> Regular text
            /   /-----------------------> Switch to reverse video & red (CSI sequence)
            /   /        /--------------> Red background & black text
            /   /        /   /----------> Reset escape sequence (CSI sequence)
            /   /        /   /     /----> Regular text
$ echo -e "xxx \e[7;31m yyy \e[0m zzz"
```



```
xxx yyy zzz
```

Finding #1

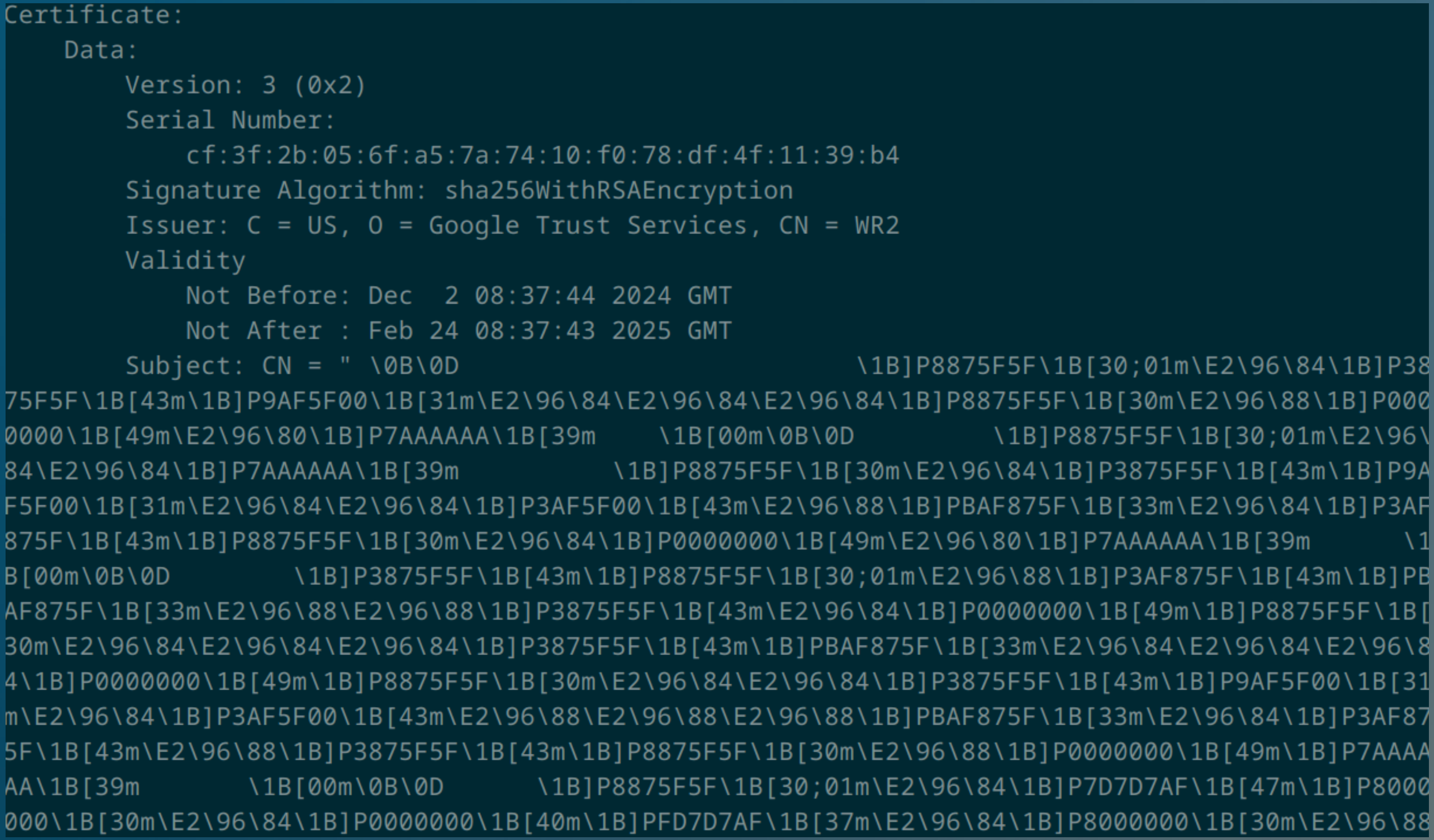
- Unicode is printed from text fields such as CN, OU, ... for recent versions of OpenSSL (>= 3.2.4 from current release set)
- As a side effect, control characters & terminal escape sequences are printed raw (unescaped)
- Works since PR #16583

```
commit 86cfd132ffc4f6198cc640a29c293850c0a59914
Date:   Sat Sep 11 13:56:28 2021 +0200

        Use -nameopt utf8 by default

 unsigned long get_nameopt(void)
 {
-    return (nmflag_set) ? nmflag : XN_FLAG_ONELINE;
+    return (nmflag_set) ? nmflag : XN_FLAG_SEP_CPLUS_SPC | ASN1_STRFLGS_UTF8_CONVERT;
```

Demo - ANSI art injected in CN

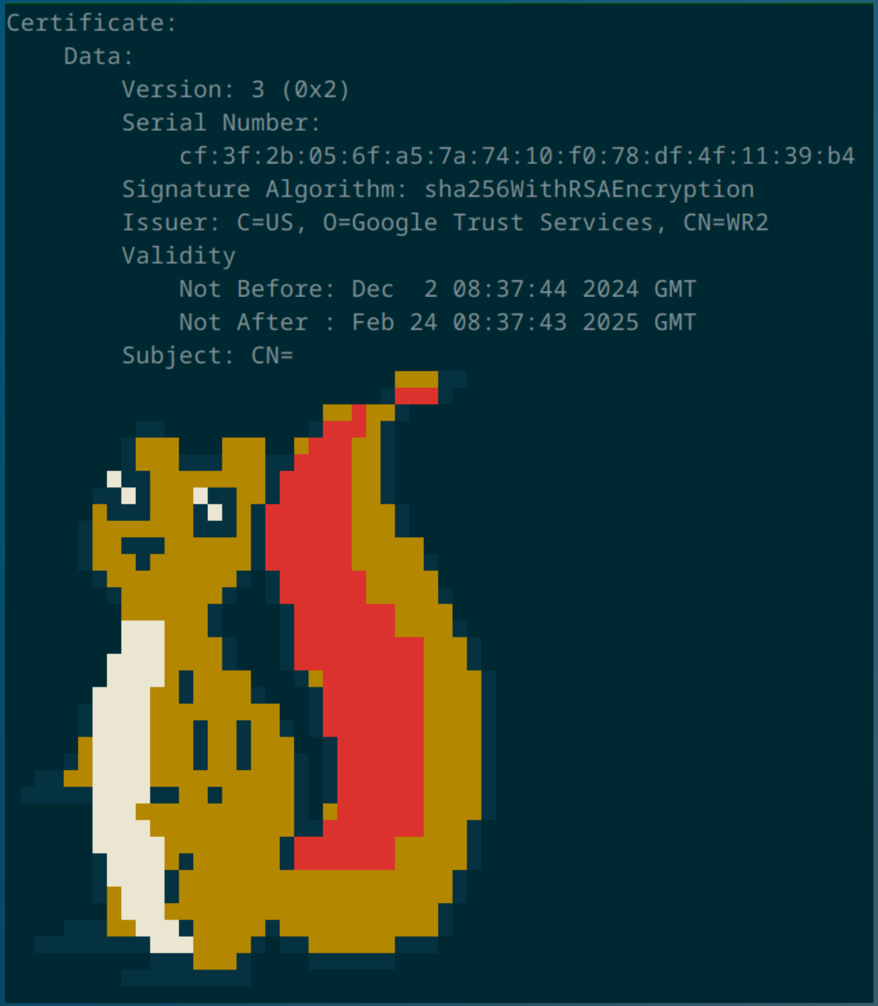
ANSI art injection in CN field with OpenSSL 3.0.16

Control characters and escape sequences are espaced correctly



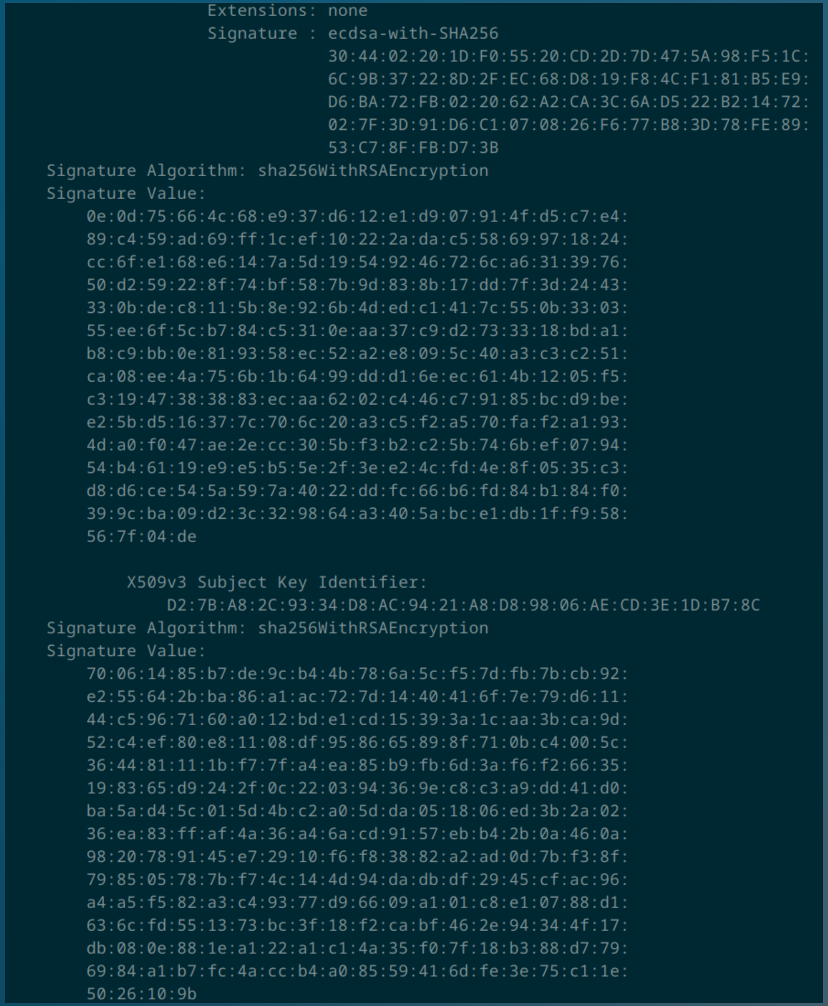
```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            cf:3f:2b:05:6f:a5:7a:74:10:f0:78:df:4f:11:39:b4
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, O = Google Trust Services, CN = WR2
        Validity
            Not Before: Dec  2 08:37:44 2024 GMT
            Not After : Feb 24 08:37:43 2025 GMT
        Subject: CN = " \0B\0D                          \1B]P8875F5F\1B[30;01m\E2\96\84\1B]P38
75F5F\1B[43m\1B]P9AF5F00\1B[31m\E2\96\84\E2\96\84\E2\96\84\1B]P8875F5F\1B[30m\E2\96\88\1B]P000
0000\1B[49m\E2\96\80\1B]P7AAAAAA\1B[39m     \1B[00m\0B\0D         \1B]P8875F5F\1B[30;01m\E2\96\
84\E2\96\84\1B]P7AAAAAA\1B[39m           \1B]P8875F5F\1B[30m\E2\96\84\1B]P3875F5F\1B[43m\1B]P9A
F5F00\1B[31m\E2\96\84\E2\96\84\1B]P3AF5F00\1B[43m\E2\96\88\1B]PBAF875F\1B[33m\E2\96\84\1B]P3AF
875F\1B[43m\1B]P8875F5F\1B[30m\E2\96\84\1B]P0000000\1B[49m\E2\96\80\1B]P7AAAAAA\1B[39m         \1
B[00m\0B\0D        \1B]P3875F5F\1B[43m\1B]P8875F5F\1B[30;01m\E2\96\88\1B]P3AF875F\1B[43m\1B]PB
AF875F\1B[33m\E2\96\88\E2\96\88\1B]P3875F5F\1B[43m\E2\96\84\1B]P0000000\1B[49m\1B]P8875F5F\1B[
30m\E2\96\84\E2\96\84\E2\96\84\1B]P3875F5F\1B[43m\1B]PBAF875F\1B[33m\E2\96\84\E2\96\84\E2\96\8
4\1B]P0000000\1B[49m\1B]P8875F5F\1B[30m\E2\96\84\E2\96\84\1B]P3875F5F\1B[43m\1B]P9AF5F00\1B[31
m\E2\96\84\1B]P3AF5F00\1B[43m\E2\96\88\E2\96\88\E2\96\88\1B]PBAF875F\1B[33m\E2\96\84\1B]P3AF87
5F\1B[43m\E2\96\88\1B]P3875F5F\1B[43m\1B]P8875F5F\1B[30m\E2\96\88\1B]P0000000\1B[49m\1B]P7AAAA
AA\1B[39m       \1B[00m\0B\0D      \1B]P8875F5F\1B[30;01m\E2\96\84\1B]P7D7D7AF\1B[47m\1B]P8000
000\1B[30m\E2\96\84\1B]P0000000\1B[40m\1B]PFD7D7AF\1B[37m\E2\96\84\1B]P8000000\1B[30m\E2\96\88
```

ANSI art injection in CN field with OpenSSL 3.4.1 - it works !



```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            cf:3f:2b:05:6f:a5:7a:74:10:f0:78:df:4f:11:39:b4
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, O=Google Trust Services, CN=WR2
        Validity
            Not Before: Dec  2 08:37:44 2024 GMT
            Not After : Feb 24 08:37:43 2025 GMT
        Subject: CN=
```

Text injection in CN field with OpenSSL 3.4.1



```
                    Extensions: none
                    Signature : ecdsa-with-SHA256
                                30:44:02:20:1D:F0:55:20:CD:2D:7D:47:5A:98:F5:1C:
                                6C:9B:37:22:8D:2F:EC:68:D8:19:F8:4C:F1:81:B5:E9:
                                D6:BA:72:FB:02:20:62:A2:CA:3C:6A:D5:22:B2:14:72:
                                02:7F:3D:91:D6:C1:07:08:26:F6:77:B8:3D:78:FE:89:
                                53:C7:8F:FB:D7:3B
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        0e:0d:75:66:4c:68:e9:37:d6:12:e1:d9:07:91:4f:d5:c7:e4:
        89:c4:59:ad:69:ff:1c:ef:10:22:2a:da:c5:58:69:97:18:24:
        cc:6f:e1:68:e6:14:7a:5d:19:54:92:46:72:6c:a6:31:39:76:
        50:d2:59:22:8f:74:bf:58:7b:9d:83:8b:17:dd:7f:3d:24:43:
        33:0b:de:c8:11:5b:8e:92:6b:4d:ed:c1:41:7c:55:0b:33:03:
        55:ee:6f:5c:b7:84:c5:31:0e:aa:37:c9:d2:73:33:18:bd:a1:
        b8:c9:bb:0e:81:93:58:ec:52:a2:e8:09:5c:40:a3:c3:c2:51:
        ca:08:ee:4a:75:6b:1b:64:99:dd:d1:6e:ec:61:4b:12:05:f5:
        c3:19:47:38:38:83:ec:aa:62:02:c4:46:c7:91:85:bc:d9:be:
        e2:5b:d5:16:37:7c:70:6c:20:a3:c5:f2:a5:70:fa:f2:a1:93:
        4d:a0:f0:47:ae:2e:cc:30:5b:f3:b2:c2:5b:74:6b:ef:07:94:
        54:b4:61:19:e9:e5:b5:5e:2f:3e:e2:4c:fd:4e:8f:05:35:c3:
        d8:d6:ce:54:5a:59:7a:40:22:dd:fc:66:b6:fd:84:b1:84:f0:
        39:9c:ba:09:d2:3c:32:98:64:a3:40:5a:bc:e1:db:1f:f9:58:
        56:7f:04:de

            X509v3 Subject Key Identifier:
                D2:7B:A8:2C:93:34:D8:AC:94:21:A8:D8:98:06:AE:CD:3E:1D:B7:8C
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        70:06:14:85:b7:de:9c:b4:4b:78:6a:5c:f5:7d:fb:7b:cb:92:
        e2:55:64:2b:ba:86:a1:ac:72:7d:14:40:41:6f:7e:79:d6:11:
        44:c5:96:71:60:a0:12:bd:e1:cd:15:39:3a:1c:aa:3b:ca:9d:
        52:c4:ef:80:e8:11:08:df:95:86:65:89:8f:71:0b:c4:00:5c:
        36:44:81:11:1b:f7:7f:a4:ea:85:b9:fb:6d:3a:f6:f2:66:35:
        19:83:65:d9:24:2f:0c:22:03:94:36:9e:c8:c3:a9:dd:41:d0:
        ba:5a:d4:5c:01:5d:4b:c2:a0:5d:da:05:18:06:ed:3b:2a:02:
        36:ea:83:ff:af:4a:36:a4:6a:cd:91:57:eb:b4:2b:0a:46:0a:
        98:20:78:91:45:e7:29:10:f6:f8:38:82:a2:ad:0d:7b:f3:8f:
        79:85:05:78:7b:f7:4c:14:4d:94:da:db:df:29:45:cf:ac:96:
        a4:a5:f5:82:a3:c4:93:77:d9:66:09:a1:01:c8:e1:07:88:d1:
        63:6c:fd:55:13:73:bc:3f:18:f2:ca:bf:46:2e:94:34:4f:17:
        db:08:0e:88:1e:a1:22:a1:c1:4a:35:f0:7f:18:b3:88:d7:79:
        69:84:a1:b7:fc:4a:cc:b4:a0:85:59:41:6d:fe:3e:75:c1:1e:
        50:26:10:9b
```

- When transposing this technique to text, what comes after the injection point is still printed
- But it does not happen with the ANSI art ?!?

- A double signature is visible in this example

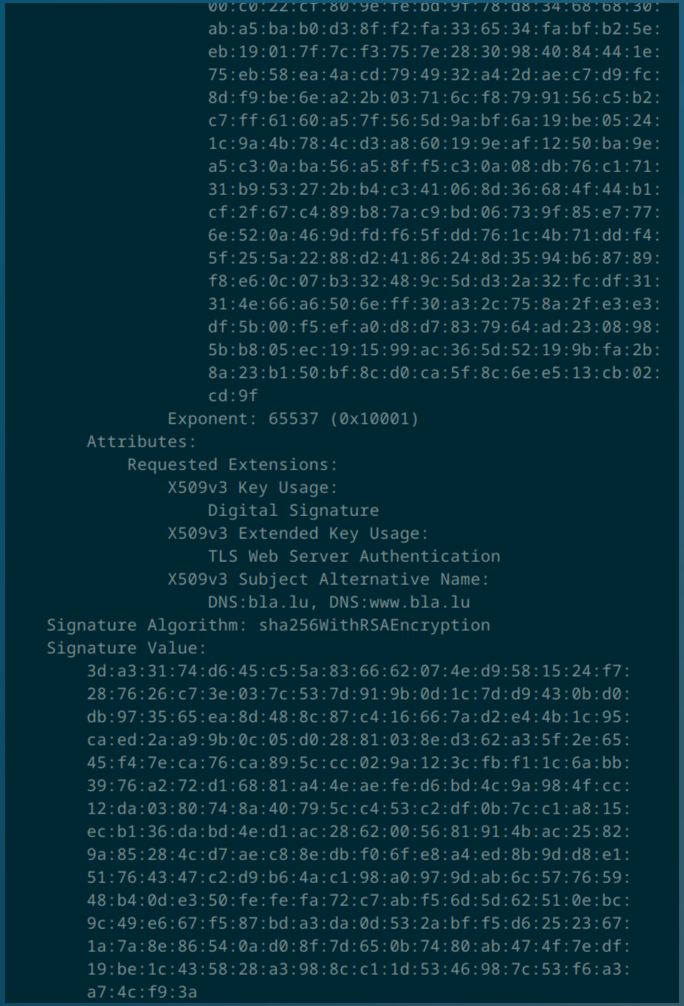
Finding #2

Unfinished OSC sequence, on Gnome Terminal (& some others) stops the text output

```
$ echo -ne "bla\nbla\n" > bla
$ echo -ne "bli\nbli\n" > bli
$ cat bla <(echo -e "\e]") bli
bla
bla
$ cat bla <(echo -e "\e]") bli | grep bli
bli
bli
```

We can use that behavior to hide text output after the injected implant

Text injection with termination in CN field with OpenSSL 3.4.1



```
                    00:c0:22:cf:80:9e:fe:bd:9f:78:d8:34:68:68:30:
                    ab:a5:ba:b0:d3:8f:f2:fa:33:65:34:fa:bf:b2:5e:
                    eb:19:01:7f:7c:f3:75:7e:28:30:98:40:84:44:1e:
                    75:eb:58:ea:4a:cd:79:49:32:a4:2d:ae:c7:d9:fc:
                    8d:f9:be:6e:a2:2b:03:71:6c:f8:79:91:56:c5:b2:
                    c7:ff:61:60:a5:7f:56:5d:9a:bf:6a:19:be:05:24:
                    1c:9a:4b:78:4c:d3:a8:60:19:9e:af:12:50:ba:9e:
                    a5:c3:0a:ba:56:a5:8f:f5:c3:0a:08:db:76:c1:71:
                    31:b9:53:27:2b:b4:c3:41:06:8d:36:68:4f:44:b1:
                    cf:2f:67:c4:89:b8:7a:c9:bd:06:73:9f:85:e7:77:
                    6e:52:0a:46:9d:fd:f6:5f:dd:76:1c:4b:71:dd:f4:
                    5f:25:5a:22:88:d2:41:86:24:8d:35:94:b6:87:89:
                    f8:e6:0c:07:b3:32:48:9c:5d:d3:2a:32:fc:df:31:
                    31:4e:66:a6:50:6e:ff:30:a3:2c:75:8a:2f:e3:e3:
                    df:5b:00:f5:ef:a0:d8:d7:83:79:64:ad:23:08:98:
                    5b:b8:05:ec:19:15:99:ac:36:5d:52:19:9b:fa:2b:
                    8a:23:b1:50:bf:8c:d0:ca:5f:8c:6e:e5:13:cb:02:
                    cd:9f
                Exponent: 65537 (0x10001)
        Attributes:
            Requested Extensions:
                X509v3 Key Usage:
                    Digital Signature
                X509v3 Extended Key Usage:
                    TLS Web Server Authentication
                X509v3 Subject Alternative Name:
                    DNS:bla.lu, DNS:www.bla.lu
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        3d:a3:31:74:d6:45:c5:5a:83:66:62:07:4e:d9:58:15:24:f7:
        28:76:26:c7:3e:03:7c:53:7d:91:9b:0d:1c:7d:d9:43:0b:d0:
        db:97:35:65:ea:8d:48:8c:87:c4:16:66:7a:d2:e4:4b:1c:95:
        ca:ed:2a:a9:9b:0c:05:d0:28:81:03:8e:d3:62:a3:5f:2e:65:
        45:f4:7e:ca:76:ca:89:5c:cc:02:9a:12:3c:fb:f1:1c:6a:bb:
        39:76:a2:72:d1:68:81:a4:4e:ae:fe:d6:bd:4c:9a:98:4f:cc:
        12:da:03:80:74:8a:40:79:5c:c4:53:c2:df:0b:7c:c1:a8:15:
        ec:b1:36:da:bd:4e:d1:ac:28:62:00:56:81:91:4b:ac:25:82:
        9a:85:28:4c:d7:ae:c8:8e:db:f0:6f:e8:a4:ed:8b:9d:d8:e1:
        51:76:43:47:c2:d9:b6:4a:c1:98:a0:97:9d:ab:6c:57:76:59:
        48:b4:0d:e3:50:fe:fe:fa:72:c7:ab:f5:6d:5d:62:51:0e:bc:
        9c:49:e6:67:f5:87:bd:a3:da:0d:53:2a:bf:f5:d6:25:23:67:
        1a:7a:8e:86:54:0a:d0:8f:7d:65:0b:74:80:ab:47:4f:7e:df:
        19:be:1c:43:58:28:a3:98:8c:c1:1d:53:46:98:7c:53:f6:a3:
        a7:4c:f9:3a
```

Finding #3

- Content appended to SAN DNS fields is copied verbatim
- Except that '\n' must be replaced by something visually equivalent such as '\v'
- This works on all version of OpenSSL I tested (>= 0.9.7m)

Demo - text injected in SAN

Real work implications

- Different results for different terminal emulators (with ASCII only, or Unicode support)
- Other SSL/TLS tookits also affected at various levels
- Also work with certificate requests . . .

Demo - Fake Certificate Request

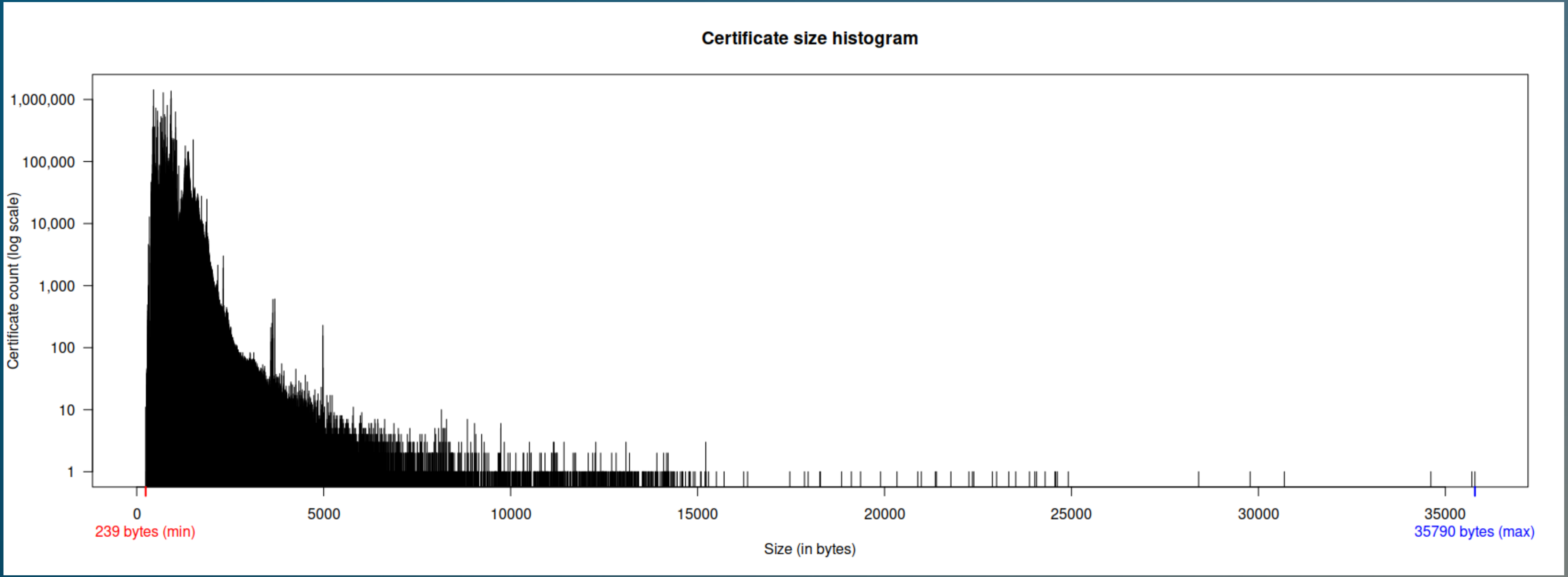
Was this technique ever used ?

Corpus of 50M+ (and counting) X.509 certificates from CIRCL's passive SSL project

circl.lu
Computer Incident
Response Center
LUXEMBOURG

https://www.circl.lu/services/passive-ssl/

Dataset size distribution histogram



Certificate oddities

```
Subject: C=UK, ST=Glasgow, L=Glasgow, O=Bellrock Technology Ltd, CN=${ansible_eth0[ipv4][address]}
```

```
Subject: C=FR, ST=Seed, L=Power, CN=${ServiceName}.alpharatio.cc
```

```
Subject: C=NG, ST=Enugu, L=ENUGU, O=Graceland Group of Schools, CN={{gracelandcloud.com}}
```

Certificate oddities

```
1.3.6.1.4.1.24255.2.1:
    {"uuid":"1b444eb0-6474-4162-a78e-5e22c76bfe00","service":"OUTSCAN"}
```

```
1.2.3.4.5.6.7.8.1:
    {"attrs":{"hf.Affiliation":"health0","hf.EnrollmentID":"ipfs0-health0","hf.Type":"ipfs"}}
```

Certificate oddities

```
Subject: C=NO, ST=molde, L=molde, O=stormtek, OU=stormtek, CN=owncloud01, emailAddress=eivind^G^G@trollhousing.no
```

```
X509v3 Authority Key Identifier:
    keyid:61:2F:38:E5:73:CC:22:6B:BE:45:25:23:0B:DF:89:7A:3F:CB:41:66
    DirName:/C=US/ST=Kentuc\x08\x08\x08\x08\x08\x08\x08\x08\x08Kentuckey\x08\x08y
        /L=Lexinghto\x08\x08\x08ton/O=PRoj\x08\x08\x08rojh\x08i
        /CN=Rex Hall/emailAddress=mmufault@otm\x08\x08\x08hotmail.com
```

Certificate oddities

```
            X509v3 Issuer Alternative Name:
                0.,^...+..+.........8....-----BEGIN PGP PRIVATE KEY BLOCK-----
Version: BCPG v1.38b04

lQPGBFcwpekBCADp7m4Z/FZHpkYPmXLo2/cuPd9wCUXFvrGas98HE0rhHhQ/apkU
TAu49tr4iV2ks7jZuP4zVA81Iob7Po+CWLD5kYcCU0PjCH3LpmFfxXiO+GhFqM4a
+KtnEFaDTi9PTBv++bhQ5ZmAAJG3Qte3Q1YTdD3uzXKEwKysSWvTGQ9RLbRp1Yo1
ZoXQzlHYinEBuWoEU2dgOeKhtvi8k52PSxbFkklf/GC0hZ+99tMnqzmS7Gl+csNJ
all/VnNoVN2JKHf8szaPMKpv2xTlStzl6lIMsdG5eswr0/Z7NqG2zsupnyn7bfPJ
[...]
```

Certificate oddities

```
            X509v3 Subject Alternative Name:
                DNESC[dESC^H^H, DNS:localhost, DNS:ip-172-31-19-174,
                    DNS:xe7p-5jtg-6snv-lne4-wfug-oaqo-hqkn-44m2-s2a2-fsqi-pvnl-iqwm,
                    IP Address:172.17.0.1, IP Address:34.252.55.170,
                    IP Address:127.0.0.1, IP Address:172.31.19.174
```

```
Subject: C='', ST=^M"><zzz=zzz>;%x\, L=^M"><zzz=zzz>;%x\, O=^M"><zzz=zzz>;%x\,
    OU=^M"><zzz=zzz>;%x\, CN=rbsec.net, emailAddress=^M"><zzz=zzz>;%x\
```

Conclusion

The decision to print terminal escape sequences should be a conscious choice made by the user

... in the mean time, you can still use less when in doubt

```
$ openssl x509 -text -noout -in certificate.pem | less
```

Bonus demo

Questions ?