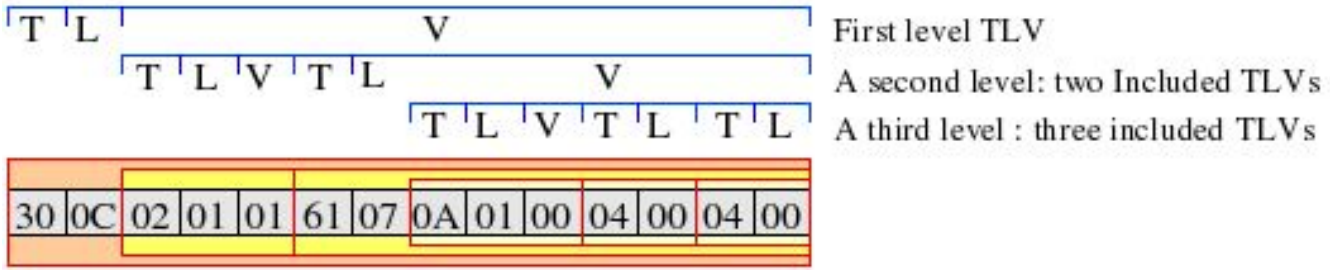
Software systems testing

Testing C# application – DER length decoder

Andronic Smaranda, Butufei Tudor, Dragomirescu Emilia, Moraru Ilinca, Tănase Daniel

ASN.1 encoding rules

- Abstract Syntax Notation One (ASN.1) is a standard of defining structure
- Mostly known because X509 certificates, used widely in TLS / HTTPS
- Encoding rules specify a TLV (tag, length, value) format



TLV encoding (source [Apache LDAP API](#))

Encoding of the length

- short form (bit 8 is 0)
  - bits 7 - 1 represent the length
- long form (bit 8 is 1)
  - bits 7 - 1 represent the number of bytes for the length (n)
  - next n bytes represent the length as an unsigned big-endian integer
- `0xxx xxxx` - data length defined by bits 7 - 1
- `1NNN NNNN xxxx xxxx ...` - `NNN NNNN` bytes following
- Examples
  - 0x03 = length 3
  - 0x82 0x01 0x03 = length 259

Tested function

```
public long GetDataLength(byte firstByte, ReadOnlySpan<byte> buffer, ref int position)
```

- Input parameters
  - **firstByte**: which is the first byte of the length byte sequence
    (it either specifies the length in short form, or the number of bytes encoding the length in long form)
  - **buffer**: buffer with potential following bytes to compute long form
  - **position**: start position in buffer to start reading from
    (value is incremented with the number of bytes used to parse the length in case of long form)
- Output
  - the length encoded by the first byte and optionally following bytes
- Thrown exceptions
  - *"Indefinite length not supported in DER."* - in case of first byte encoding indefinite length
  - *"Invalid position"* - in case first byte encodes long form and provided position parameter is out of bounds for buffer
  - *"Unexpected end of data"* - in case not enough bytes in buffer starting from position to compute encoded length

Tested function

```
public long GetDataLength(byte firstByte, ReadOnlySpan<byte> buffer, ref int position)
{
    // Check if single byte length
    if ((firstByte & 0x80) == 0)
    {
        return firstByte;
    }

    // Get the number of bytes that compose the length
    int numBytes = firstByte & 0x7F;
    if (numBytes == 0)
    {
        throw new Exception("Indefinite length not supported in DER.");
    }

    // Validate parameter position
    if (position < 0 || position >= buffer.Length)
    {
        throw new Exception("Invalid position");
    }

    // Add each byte to the length
    long length = 0;
    for (int i = 0; i < numBytes; i++)
    {
        if (position < buffer.Length)
        {
            length = (length << 8) | buffer[position++];
        }
        else
        {
            throw new Exception("Unexpected end of data");
        }
    }

    return length;
}
```

Equivalence classes

Why equivalence classes?

In testing, equivalence classes represent partitioning the inputs into groups, were we assume the program would have similar behavior for all the elements in a group.

Equivalence classes

1. First byte

- **F_1: 0x00 - 0x7F** (first bit 0, representing short form length)
- **F_2: 0x80** (indefinite length, not allowed in DER)
- **F_3: 0x81 - 0xFF** (first bit 1, representing number of following bytes encoding long form)

3. Following Bytes (length value bytes)

- **B_1: Too few (< declared)** → Error: insufficient data
- **B_2: Enough (≥ declared)** → Valid: full length available

3. Offset position

- **P_1: < 0** (invalid array index)
- **P_2: ≥ and < buffer length** (valid array index)
- **P_3: ≥ buffer length** (invalid array index)

Equivalence classes

Case	First_byte	Buffer	Position	Interpretation
C_1	F_1	___	___	Returns length from short form encoding
C_2	F_2	___	___	Indefinite length not supported in DER exception
C_3_1	F_3	___	P_1	Invalid position exception
C_3_2	F_3	___	P_3	Invalid position exception
C_3_3_1	F_3	B_1	P_3	Returns length from long form encoding
C_3_3_2	F_3	B_2	P_3	Unexpected end of data exception

Boundary Value Analysis

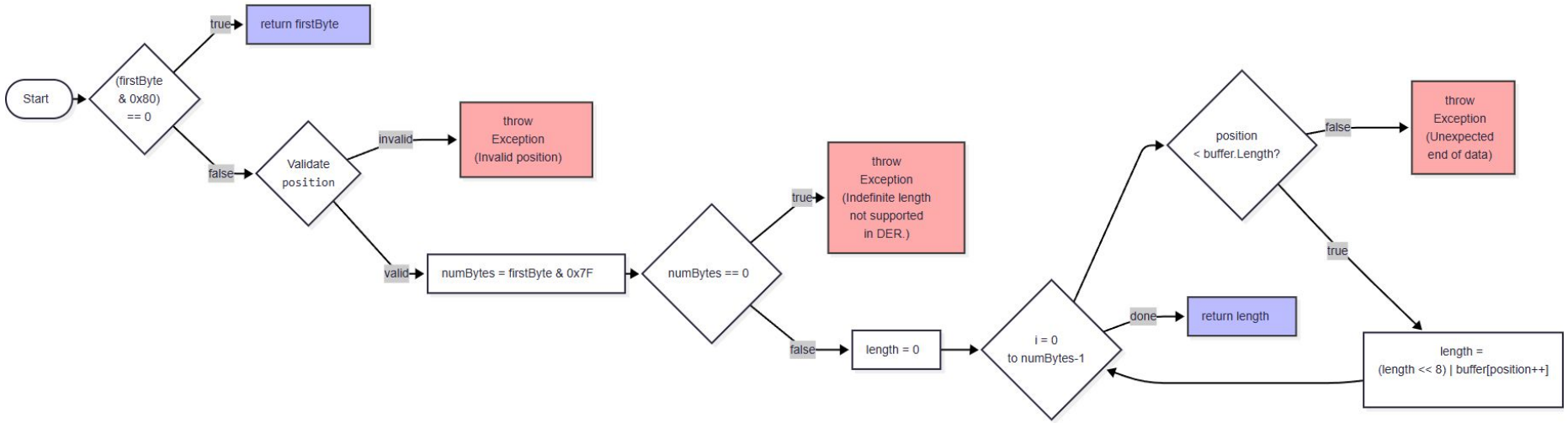
We use the same classes: F, B and P. Next, we look for the boundary values of each class.

- First byte:
  - 0x00 (BIT1: 0, BIT2-8: 0), 0x01 (BIT1: 0, BIT2-8: 1), 0x7F (BIT1: 0, BIT2-8: 127), 0x80 (BIT1: 1, BIT2-8: 0), 0x81 (BIT1: 1, BIT2-8: 1), 0xFF (BIT1: 1, BIT2-8: 127)
- The following bytes buffer length: can be between 1 and 127 or none. We test N/A, 1, 127, 128
- Position:
  - -1, 0, and buffer length

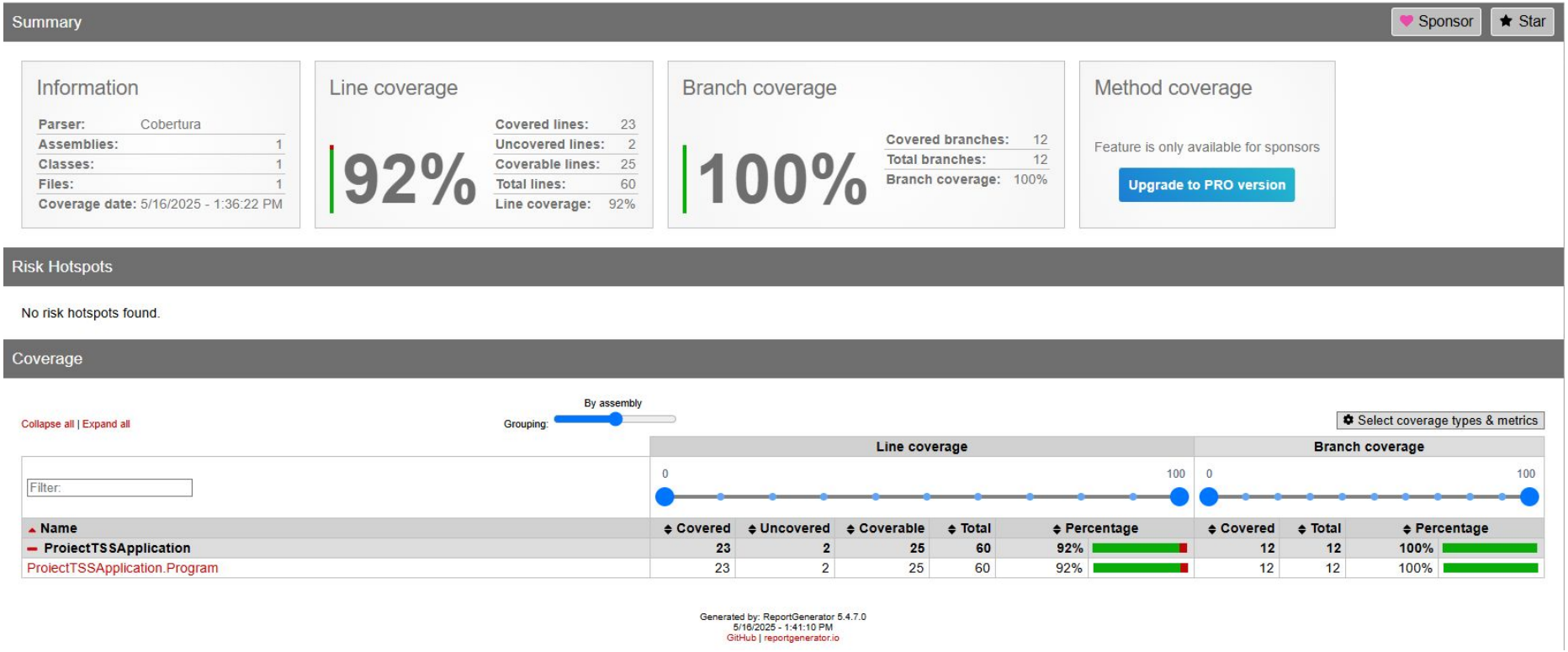
Boundary Value Analysis

Case	First byte	Payload Bytes	Position	Expected Behavior
C_1_1	0x00	---	0	Valid – Result = 0, the rest ignored
C_1_2	0x01	---	0	Valid – Result = 1, the rest ignored
C_1_3	0x7F	---	0	Valid – Result = 127, the rest ignored
C_2_1	0x80	---	0	Invalid – Length = 0 not allowed
C_2_2_1	0x81	N/A	0	Invalid – 1 byte expected, got none
C_2_2_2	0x81	F0	0	Valid – Result = 240(0xF0) exact match
C_2_2_3	0x81	A0 x 127	0	Valid – Result = 160(0xA0) extra bytes allowed
C_2_2_4	0x81	A0 x 128	0	Valid – Result = 160(0xA0) extra bytes allowed
C_2_3_1	0xFF	N/A	0	Invalid – 127 bytes expected, got none
C_2_3_2	0xFF	F0	0	Invalid – only 1 byte, 126 missing
C_2_3_3	0xFF	A0 x 127	0	Valid – Result = 0xA0 x 127 exact match
C_2_3_4	0xFF	A0 x 128	0	Valid – Result = 0xA0 x 127 extra byte allowed
C_2_4_1	0x81	F0	-1	Invalid - -1 is not a valid index
C_2_4_2	0x81	F0	1	Invalid - 1 is not a valid index
C_2_4_3	0xFF	A0 x 127	-1	Invalid - -1 is not a valid index
C_2_4_4	0xFF	A0 x 127	127	Invalid - 127 is not a valid index

Structural testing



Structural testing



Summary

Sponsor Star

Information

Parser:	Cobertura
Assemblies:	1
Classes:	1
Files:	1
Coverage date:	5/16/2025 - 1:36:22 PM

Line coverage

92%

Covered lines:	23
Uncovered lines:	2
Coverable lines:	25
Total lines:	60
Line coverage:	92%

Branch coverage

100%

Covered branches:	12
Total branches:	12
Branch coverage:	100%

Method coverage

Feature is only available for sponsors

Upgrade to PRO version

Risk Hotspots

No risk hotspots found.

Coverage

Collapse all | Expand all

Grouping: By assembly

Select coverage types & metrics

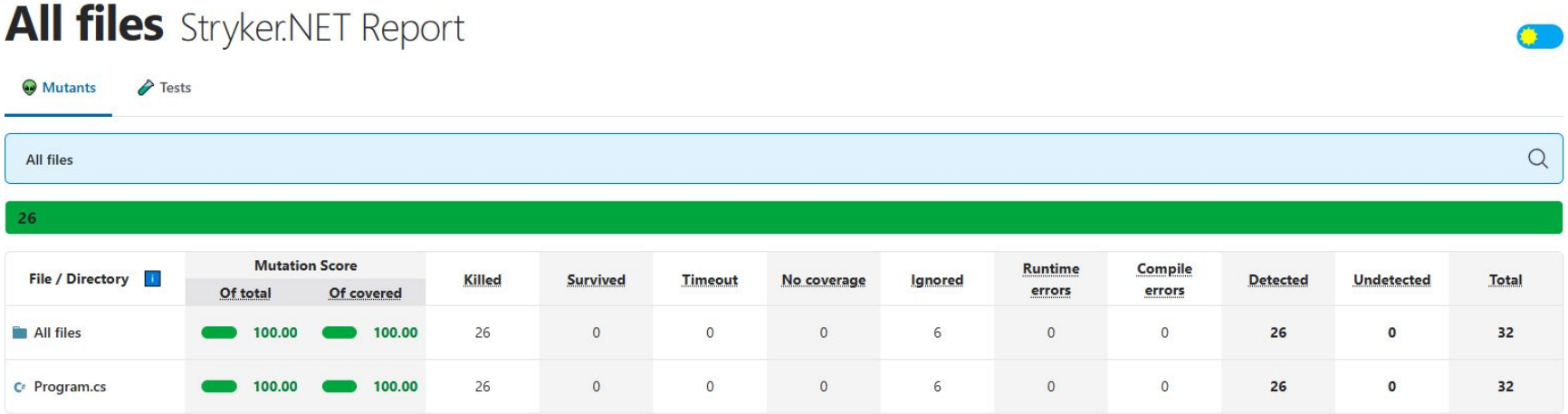
Filter:

Name	Line coverage: Covered	Uncovered	Coverable	Total	Percentage	Branch coverage: Covered	Total	Percentage
ProiectTSSApplication	23	2	25	60	92%	12	12	100%
ProiectTSSApplication.Program	23	2	25	60	92%	12	12	100%

Generated by: ReportGenerator 5.4.7.0
5/16/2025 - 1:41:10 PM
GitHub | reportgenerator.io

Mutation testing

- We used Stryker tool to generate mutants and test them against the tests
- Initially we had 7 surviving because of unrelated code



All files Stryker.NET Report

Mutants | Tests

All files

26

File / Directory	Mutation Score		Killed	Survived	Timeout	No coverage	Ignored	Runtime errors	Compile errors	Detected	Undetected	Total
	Of total	Of covered										
All files	100.00	100.00	26	0	0	0	6	0	0	**26**	**0**	**32**
Program.cs	100.00	100.00	26	0	0	0	6	0	0	**26**	**0**	**32**