PA193 Secure coding principles and practices: XML Parser report

Matúš Nemec, Pankaj Agarwal, Anupam Gupta (Team B)

The XML parser is intended to work in a document validation mode. Document processing will halt at first error and provide an informative error code. The parser is building an abstract representation of the document in the memory, based on a rooted tree. For demonstration purposes, correctly parsed document is printed without being manipulated.

Rules for valid documents are based on the XML specification. Some modifications were required, in order to reduce the scope of specification for our purpose. For full list of rules, please refer to specification delivered with the parser source code. Features of the parser can be divided into three main categories:

1. Supported – these document structures are handled exactly as required or very closely to the real specification.
   - Properly nested tags, which define elements.
   - Attributes in tags, which must have unique names.
   - Properly formed comments placed at allowed places.
2. Partially supported – these structures are supported, because they are usually included in an XML document. However their content will be ignored or conformance to specification will not be enforced.
   - Document declaration – values do not play any role and only proper format and placement is enforced.
3. Not supported – these constructs are part of the XML specification, however they are outside of our scope and are in most cases considered as errors.
   - Processing instructions and other special constructs are not allowed in our specification. If some of these can be "mistaken" for a more general case in our model, then they will not be processed as defined by official specification.

The implementation is using only standard libraries and no code of a third person was used. Parsing method was designed without referring to other implementations as well. XML specification states which characters are allowed or prohibited in different places. These information provide small hints, that were useful when making our parsing method. The parser contains 500 lines of C++ code and 90 lines in header file. Comments add additional 200 lines.

Building blocks of an XML document are tags, therefore in the lowest layer, we are parsing tags and pairing them to form elements in a higher layer. Tags begin the same way as comments (which are ignored) and are processed by the same function. Valid tags are empty element tags, begin and end tags. Document declaration is parsed separately. Text content between tags is also allowed and is saved to relevant element. Begin tags and empty elements may contain attributes. Mapping the attribute name to the value is used. Map provides efficient name uniqueness check.

Restrictions were introduced only when necessary. Some characters in tag and attribute names are forbidden by official specification, therefore they are not accepted. Some control characters are allowed even when not playing the control role, if their presence cannot be mistaken for special meaning. White spaces are also allowed matching the specification closely.

Selected rules (for more rules, please refer to our scope specification):

- Syntax character "<" must not appear anywhere except when performing its markup-delineation role.
- There is a single "root" element.
- The <begin>, </end>, and <empty-element/> tags that delimit the elements are correctly nested, with none missing and none overlapping. Names of the elements are case sensitive.
- Content must not contain "<" (general rule) and also ">" is not allowed.
- Comment may contain any sequence of characters except for "– –" (reserved).