XML - Processing Modules

Python library description

Marian Petruk

April 27 2017

Contents

1 Definition(s) **2**

2 Python XML processing modules **2**

2.1 Intro 2

2.2 XML vulnerabilities 3

2.3 XML submodules 3

2.4 Secure modules 3

3 XML Python library usage **4**

1 Definition(s)

1. ***Extensible Markup Language (XML)*** *is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The W3C's XML 1.0 Specification and several other related specifications—all of them free open standards—define* ***XML****. The design goals of* ***XML*** *emphasize simplicity, generality, and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, the language is widely used for the representation of arbitrary data structures such as those used in web services.*
   Several schema systems exist to aid in the definition of ***XML****-based languages, while programmers have developed many application programming interfaces (APIs) to aid the processing of XML data. - Wikipedia, the free encyclopedia.*

2. ***XML*** *is a software- and hardware-independent tool for storing and transporting data.*
   What is XML?

   (a) XML stands for eXtensible Markup Language

   (b) XML is a markup language much like HTML

   (c) XML was designed to store and transport data XML was designed to be self-descriptive

   (d) XML is a W3C Recommendation

   - w3schools.com, the world's largest web developer site.

2 Python XML processing modules

2.1 Intro

Python 3.5.3 documentation - xml

XML interfaces for Python programming language are grouped in the xml package.

A little disclaimer in the beggining:
Nowadays JSON is more popular than XML due to some issues.

Warning

The XML modules are not secure against erroneous or maliciously constructed data.
XML is much more difficult to parse than JSON.

The documentation for the xml.dom and xml.sax packages are the definition of the Python bindings for the DOM and SAX interfaces.

Table 1: Common attacks and vulnerability of the modules

kind	sax	etree	minidom	pulldom	xmlrpc
billion laughs	**Vulnerable**	**Vulnerable**	**Vulnerable**	**Vulnerable**	**Vulnerable**
quadratic blowup	**Vulnerable**	**Vulnerable**	**Vulnerable**	**Vulnerable**	**Vulnerable**
external entity expansion	**Vulnerable**	Safe	Safe	**Vulnerable**	Safe
DTD retrieval	**Vulnerable**	Safe	Safe	**Vulnerable**	Safe
decompression bomb	Safe	Safe	Safe	Safe	**Vulnerable**

2.2 XML vulnerabilities

2.3 XML submodules

The main XML submodules are:

- xml.etree.ElementTree: the ElementTree API, a simple and lightweight XML processor for parsing and creating xml data
- xml.dom: the DOM API definition. DOM or Document Object Model is a cross-language API for accessing and modigying XML documents. It provides a model of a document as a tree structure. So it is very easy to access the structure elements.
- xml.dom.minidom: a minimal DOM implementation. The main goal of minidom is to be simpler than normal full DOM, it is also significantly smaller.
- xml.dom.pulldom: support for building partial DOM trees. It provides so called "full parser" which can also produce DOM-accessible fragments of the document. The main idea is to pull "events" from a stream of incoming XML and process them.
- xml.sax: SAX2 base classes and convenience functions. This package consists of a number of modules for the Simple API for XML (SAX) interface for Python.
- xml.parsers.expat: the Expat parser binding. This package is an interface to the Expat non-validating XML parser. It provides an extension *xmlparser*, that represents the current state of an XML parser.

2.4 Secure modules

The *defusedxml* and *defusedexpat* packages.

1. defusedxml is a package with modified subclasses of all stdlib XML parsers for preventing any potentially malicious operation. It is recomended to use it for any code that parses untrusted XML data. It also provides example exploits and extended documentation on more XML exploits e.g. *XPath injection*.
2. defusedxpat provides a modified *libexpat* and a patched *pyexpat* module that have can resist against entity expansion DoS attacks.

3 XML Python library usage

```
$ wget http://www.diveintopython3.net/examples/feed.xml
```

```
>>> import xml.etree.ElementTree as etree
>>> tree = etree.parse('example/feed.xml')
>>> root = tree.getroot()
>>> root
<Element '{http://www.w3.org/2005/Atom}feed' at 0x7fb971161bd8>
>>> root.tag
'{http://www.w3.org/2005/Atom}feed'

>>> len(root)
8
>>> for child in root:
...     print(child)
...
<Element '{http://www.w3.org/2005/Atom}title' at 0x7fb971178188>
<Element '{http://www.w3.org/2005/Atom}subtitle' at 0x7fb97061c5e8>
<Element '{http://www.w3.org/2005/Atom}id' at 0x7fb97062ed68>
<Element '{http://www.w3.org/2005/Atom}updated' at 0x7fb97062edb8>
<Element '{http://www.w3.org/2005/Atom}link' at 0x7fb97062ee58>
<Element '{http://www.w3.org/2005/Atom}entry' at 0x7fb97062eea8>
<Element '{http://www.w3.org/2005/Atom}entry' at 0x7fb97016b3b8>
<Element '{http://www.w3.org/2005/Atom}entry' at 0x7fb97016b778>

>>> root.attrib
{'{http://www.w3.org/XML/1998/namespace}lang': 'en'}
>>> root[4]
<Element '{http://www.w3.org/2005/Atom}link' at 0x7fb97062ee58>
>>> root[4].attrib
{'rel': 'alternate', 'type': 'text/html', 'href': 'http://diveintomark.org/'}

>>> tree.findall('{http://www.w3.org/2005/Atom}entry')
[<Element '{http://www.w3.org/2005/Atom}entry' at 0x7fb97062eea8>, <Element
↪  '{http://www.w3.org/2005/Atom}entry' at 0x7fb97016b3b8>, <Element
↪  '{http://www.w3.org/2005/Atom}entry' at 0x7fb97016b778>]

>>> import xml.etree.ElementTree as etree
>>> new_feed = etree.Element('{http://www.w3.org/2005/Atom}feed',
...     attrib={'{http://www.w3.org/XML/1998/namespace}lang': 'en'})
>>> print(etree.tostring(new_feed))
b'<ns0:feed xmlns:ns0='http://www.w3.org/2005/Atom' xml:lang='en'/>'
```