XPath

Challenges

Notes

XML (E**X**tensible **M**arkup **L**anguage)

a markup language (such as HTML) mainly designed to describe data and not to display it. Due to their nature, XML documents are often used as databases. Data can be read and written through queries, and the XML database looks just like an XML document.

example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>                                   --->> defines the
current version number
<users>
          <user id='1'>
               <username>jason</username>
               <!-- Comment -->
               <password>dh8Gjnkj</password>
          </user>
          <user id='2'>
               <username>chris</username>
               <password>ddssdPikmd</password>
          </user>
</users>
```

XPath (XML Path Language) is a standard language used to query and navigate XML documents ,it makes use of path expressions to select nodes from an XML document .

//user   --> select all users
/users//username   --> Select all username elements that are a child of the users element
//user[@id='1']/username -->Select the username elements that have the parent user id attribute value set to 1.
//user[username/text()='<USERNAME>' and password/text()='<PASSWORD>']

- //: select all user elements no matter where they are in the document
- username/text()='<USERNAME>': return only the element with the username text value set to <USERNAME>
- and: Boolean operator
- password/text()='<PASSWORD>': return only the element with the password text value set to <PASSWORD>
- Where <USERNAME> and <PASSWORD> are input values

- /: select the document node
    - i.e. /root will select the root element
- //: select all nodes (that match the selection) regardless of their position in the document
    - i.e. users//user select all user elements, no matter where they are under the users element
- node_name: select all nodes with name node_name
    - i.e. users/user select all user elements that are children of users
- @: select attributes
    - i.e., /@id select all attributes that are named id

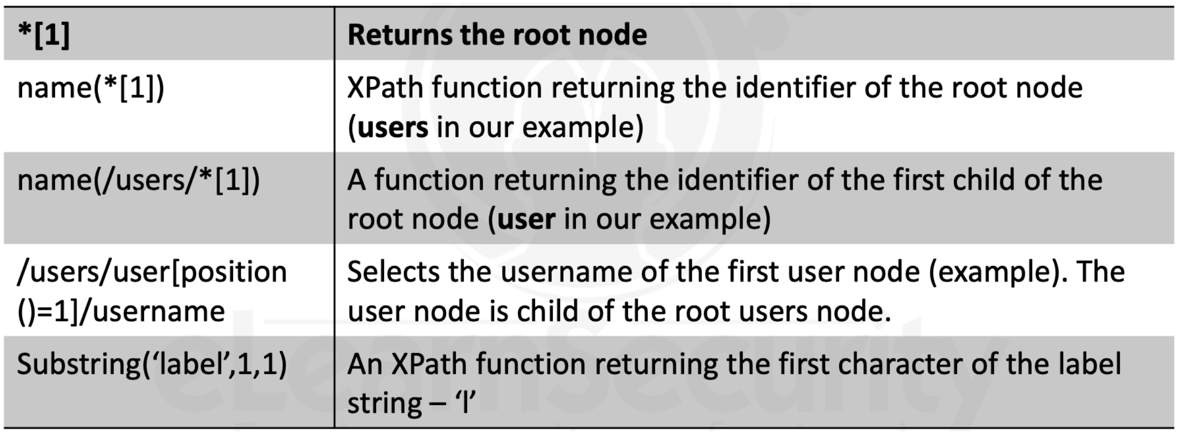
- [element and condition]: select all nodes that match the defined
  - i.e., user[username] select all user elements that contain at least one username element child
  - i.e., user[username/text()='john'] select all user elements that contain the username element child text set to 'john'
  - i.e., //user[@id='1'] select all user elements (no matter where they are in the document), with the attribute id set to 1

the XPath language does not permit comment expressions so no # or -- -
XPath is a case-sensitive language
logical operators : and , or , not()

for detection try ' and comma

Xpath exploited for :

- Bypassing authentication
  - //<someNode>[username='' or 'a'='a' or 'a'= 'a' and password='']   --> these are singe quotes not double
  - can be seen as(username='' or 'a'='a') or ('a'= 'a' and password='') --> first condition always true
- Extracting the XML document structure and contents



***[1]**	**Returns the root node**
name(*[1])	XPath function returning the identifier of the root node (**users** in our example)
name(/users/*[1])	A function returning the identifier of the first child of the root node (**user** in our example)
/users/user[position ()=1]/username	Selects the username of the first user node (example). The user node is child of the root users node.
Substring('label',1,1)	An XPath function returning the first character of the label string – 'l'

Blind injection

' or substring(name(/*[1]),1,1)= 'a
' or substring(name(/*[1]),1,1)= 'b
' or substring(name(/*[1]),1,1)= 'c

/users/user[position()==$i]/username

best way to protect against XPath injection is to filter all input data.
filter out any non- alphanumerical character.

example:
• $username=filterChars($_GET['username']);
• $password=filterChars($_GET['password']);
• $query="//user[username/text()='".$username."' and password/text()='".$password."']/username";

Tools

https://github.com/orf/xcat
https://code.google.com/archive/p/xpath-blind-explorer/