

# XPath Injection

## 1. XPath Injection Description

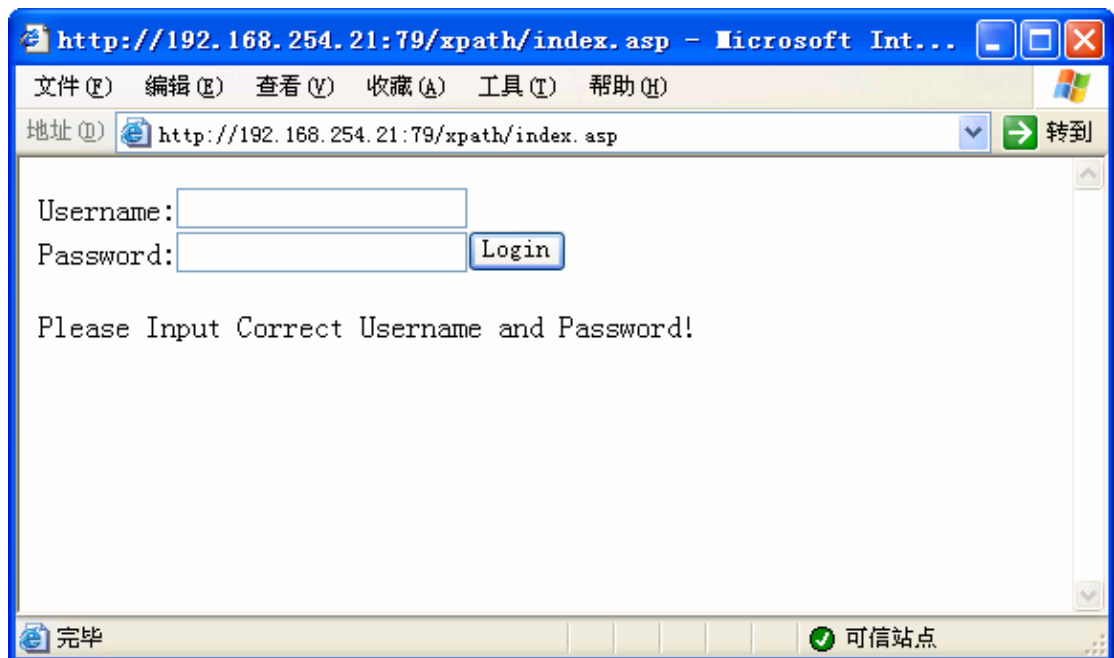
Similar to SQL Injection, XPath Injection attacks occur when a web site uses user-supplied information to construct an XPath query for XML data. By sending intentionally malformed information into the web site, an attacker can find out how the XML data is structured, or access data that he may not normally have access to. He may even be able to elevate his privileges on the web site if the XML data is being used for authentication (such as an XML based user file).

Querying XML is done with XPath, a type of simple descriptive statement that allows the XML query to locate a piece of information. Like SQL, you can specify certain attributes to find, and patterns to match. When using XML for a web site it is common to accept some form of input on the query string to identify the content to locate and display on the page. This input must be sanitized to verify that it doesn't mess up the XPath query and return the wrong data.

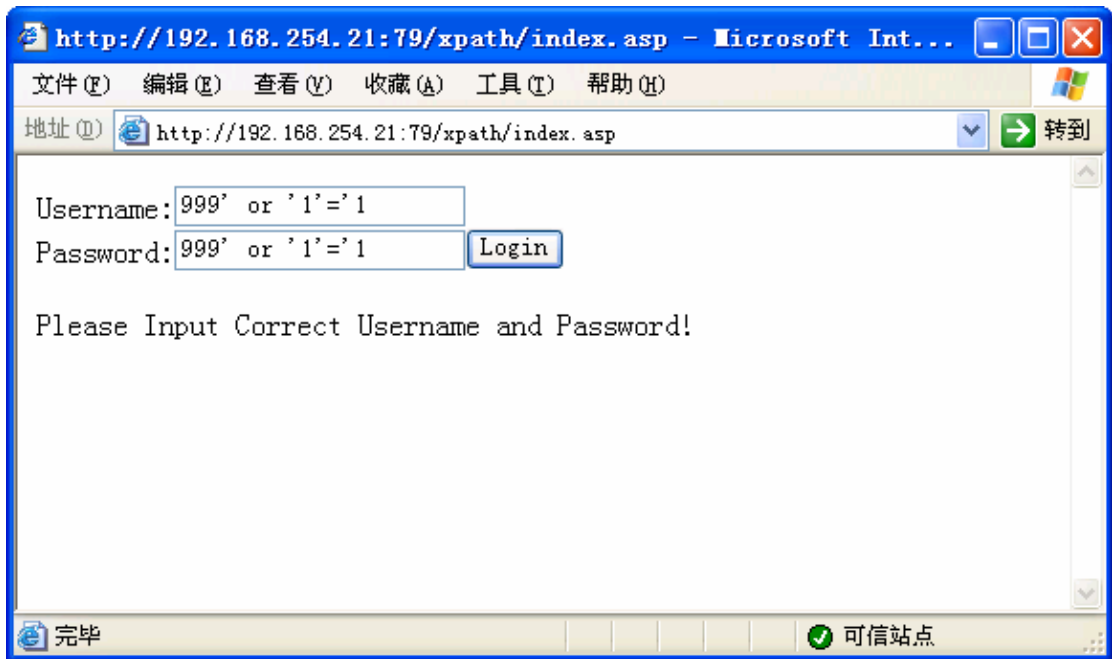
XPath is a standard language; its notation/syntax is always implementation independent, which means the attack may be automated. There are no different dialects as it takes place in requests to the SQL databases.

Because there is no level access control it's possible to get the entire document. We won't encounter any limitations as we may know from SQL injection attacks.

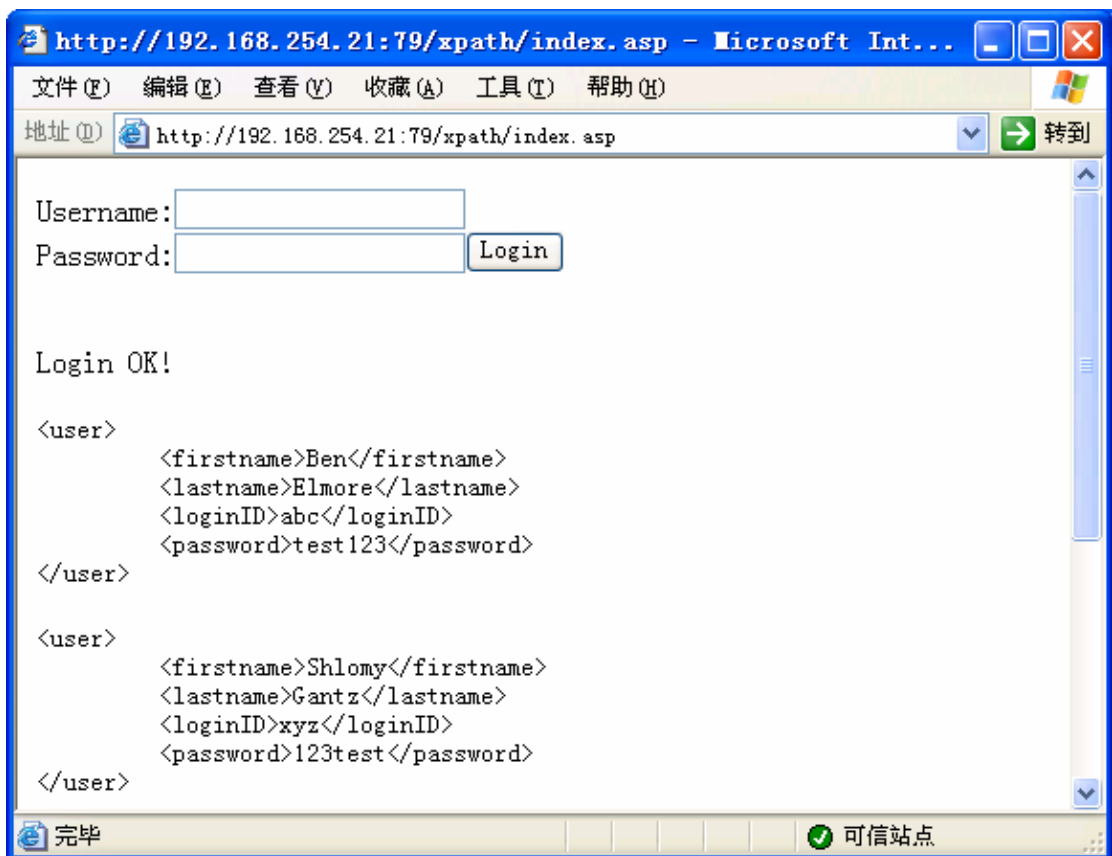
Example:



Input:



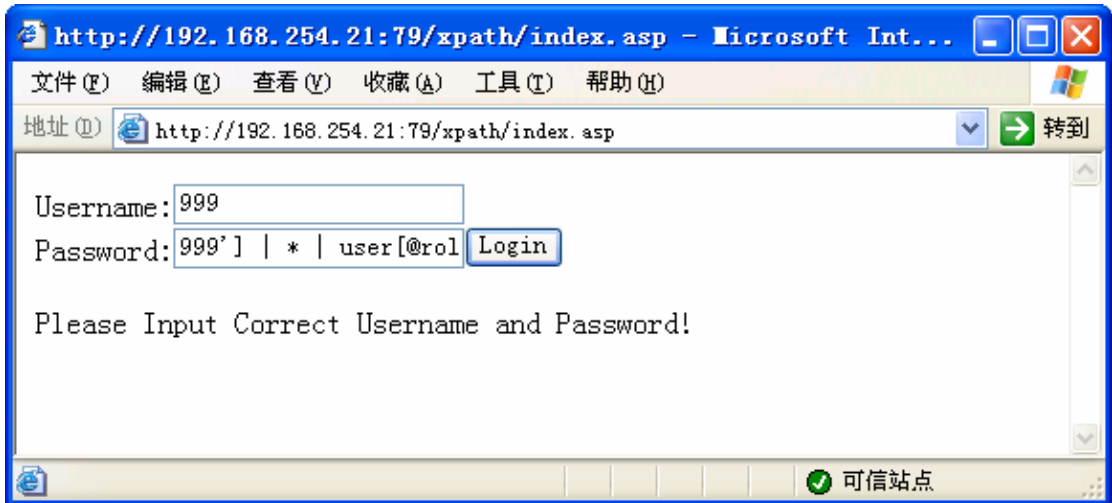
Result:



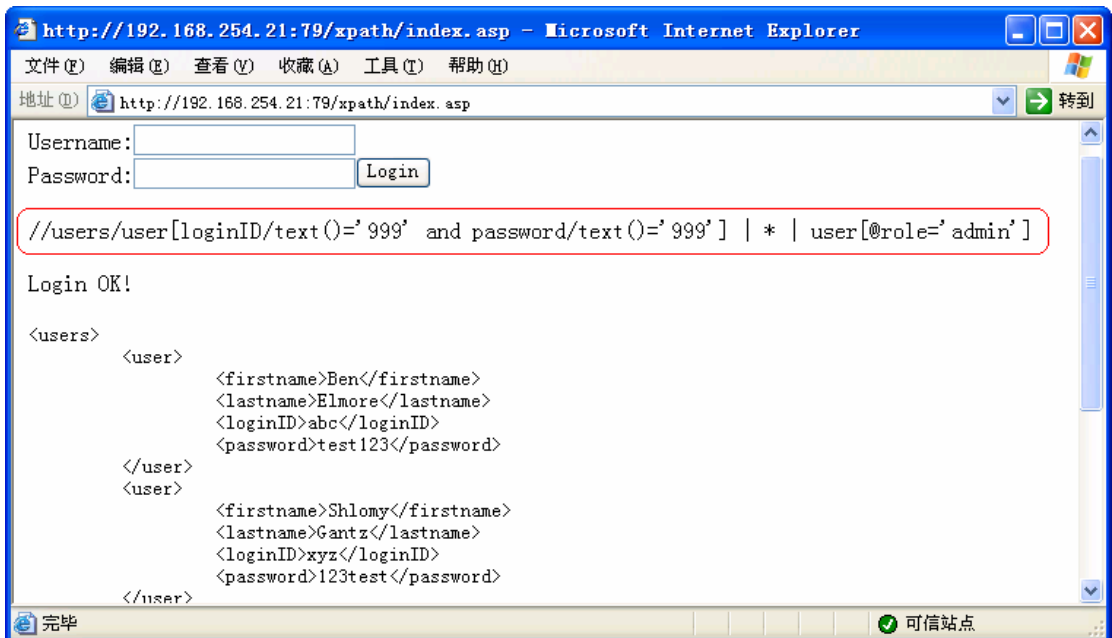
The result includes much sensitive information, now you can get a conclusion that the application use XML file to store user authentication data.

In order to analyze the injection process, we modify the sever script to output the query sentence to user's browser. Input the following username or password:

999'] | \* | user[@role='admin



Result:



The text with red frame is the XPath query sentence. `999' | * | user[@role='admin'` has been injected the sentence successfully.

Now, let's see the source code of index.asp:

```
<script language="javascript" runat="server">
Response.write("<html><body>");
uid=Request.form("uid");
pwd=Request.form("pwd");
Response.write("<form          method='POST'\>Username:<input          name='uid\'
size='20\'/><br>Password:<input  name='pwd\'  size='20\'/><input  type='submit\'
value='Login\'/></form>");
var xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async="false";
xmlDoc.load("/Inetpub/wwwroot/xpath/user.xml");
var auth="//users/user[loginID/text()='"+uid+"' and password/text()='"+pwd+"']";
```

```

Response.write(auth);
var UserObj=xmlDom.selectNodes(auth);
if(UserObj.length>0) Response.write("<br><br>Login OK!");
else Response.write("Please Input Correct Username and Password!");
Response.write(UserObj.Xml);
for(var i=0;i<UserObj.length;i++)
{
    Response.write("<xmp>");
    Response.write(UserObj(i).xml);
    Response.write("</xmp>");
}
Response.write("</body></html>");
</script>

```

user authentication file user.xml :

```

<?xml version="1.0" encoding="UTF-8"?>
<users>
    <user>
        <firstname>Ben</firstname>
        <lastname>Elmore</lastname>
        <loginID>abc</loginID>
        <password>test123</password>
    </user>
    <user>
        <firstname>Shlomy</firstname>
        <lastname>Gantz</lastname>
        <loginID>xyz</loginID>
        <password>123test</password>
    </user>
</users>

```

You can get the XPath query sentence as follow:

`auth="//users/user[loginID/text()=''+uid+"' and password/text()=''+pwd+"']"`

It means that, select user nodes which uid is equal to your input uid and password is equal to your input pwd;

The actual XPath sentence is set to:

`//users/user[loginID/text()='999' and password/text()='999'] | * | user[@role='admin']` ,

The logic result is select all nodes, XPath injection occurred.

## 2. XPath Injection Tool

WebCruiser - Web Vulnerability Scanner

WebCruiser - Web Vulnerability Scanner, a compact but powerful web security scanning tool!  
It has a Crawler and Vulnerability Scanner(SQL Injection, Cross Site Scripting, XPath Injection etc. ).

It can support scanning website as well as POC( Proving of concept) for web vulnerabilities: SQL Injection, Cross Site Scripting, XPath Injection etc. So, WebCruiser is also a SQL Injector, a XPath Injector , and a Cross Site Scripting tool!

Function:

- \* Crawler(Site Directories And Files);
- \* Vulnerability Scanner(SQL Injection, Cross Site Scripting, XPath Injection etc.);
- \* POC(Proof of Concept): SQL Injection, Cross Site Scripting, XPath Injection etc.;
- \* GET/Post/Cookie Injection;
- \* SQL Server: PlainText/FieldEcho(Union)/Blind Injection;
- \* MySQL/Oracle/DB2/Access: FieldEcho(Union)/Blind Injection;
- \* Administration Entrance Search;
- \* Time Delay For Search Injection;
- \* Auto Get Cookie From Web Browser For Authentication;
- \* Report Output.

<http://sec4app.com>