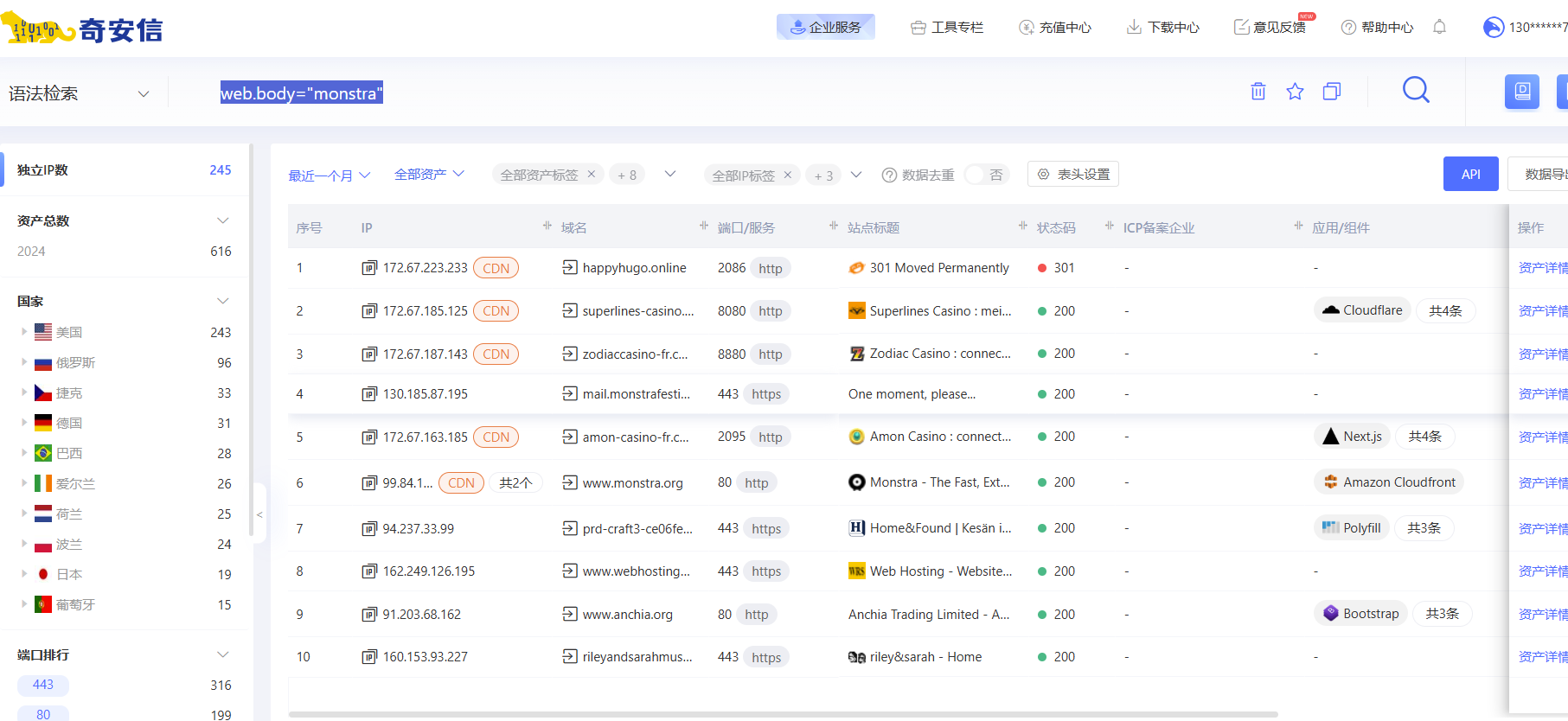
**漏洞描述**

Monstra cms 在路径/monstra-3.0.4/users/5/edit中存在**存储型**xss漏洞

源码地址：https://github.com/monstra-cms/monstra/

源码归属：https://monstra.org/

hunter搜索web.body="monstra"



互联网案例：

https://ridgewayscoaches.co.uk/home,https://mail.ridgewayscoaches.co.uk/,https://rileyandsarahmusic.com/home,https://demo.harmonyb2b.io/,https://52.58.6.138/

存储型XSS——/monstra-3.0.4/users/1/edit

漏洞详情

Monstra CMS最先版本3.0.4存在存储型XSS漏洞，攻击者可以通过/monstra-3.0.4/users/1/edit中About Me 的输入框进行XSS注入

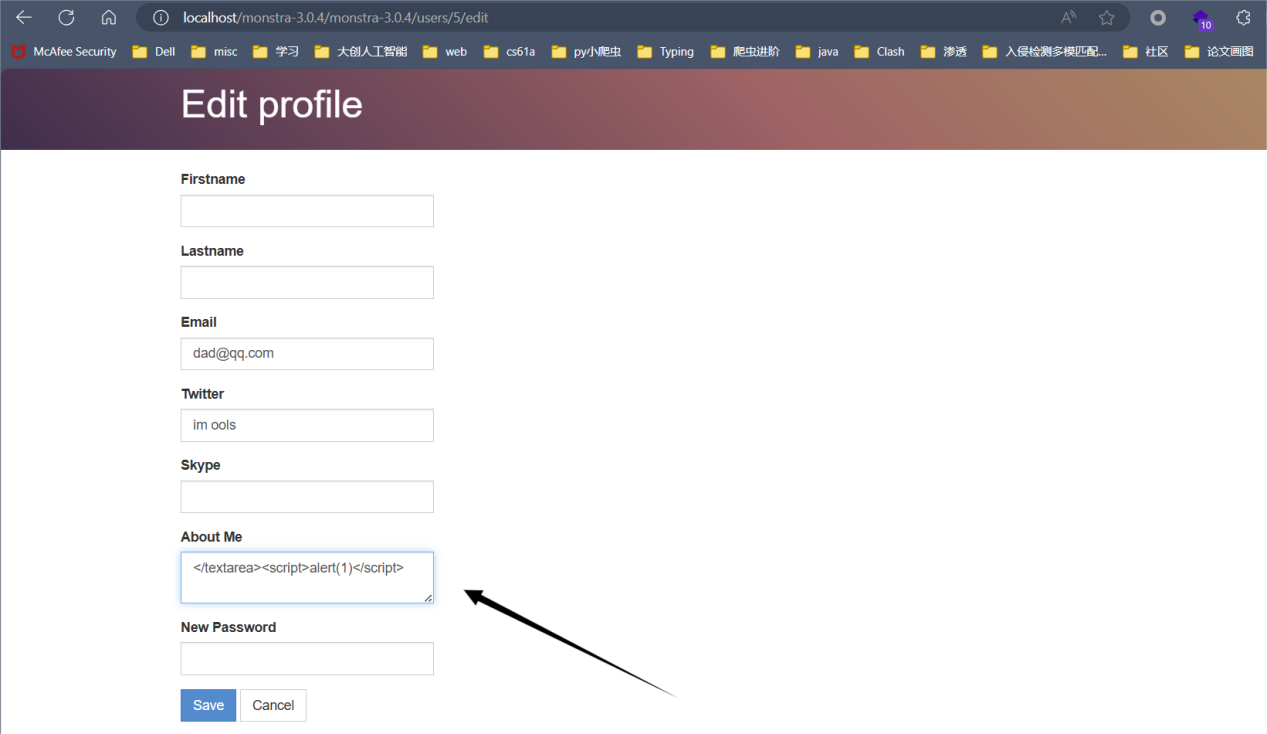
漏洞复现

漏洞环境搭建在本地。

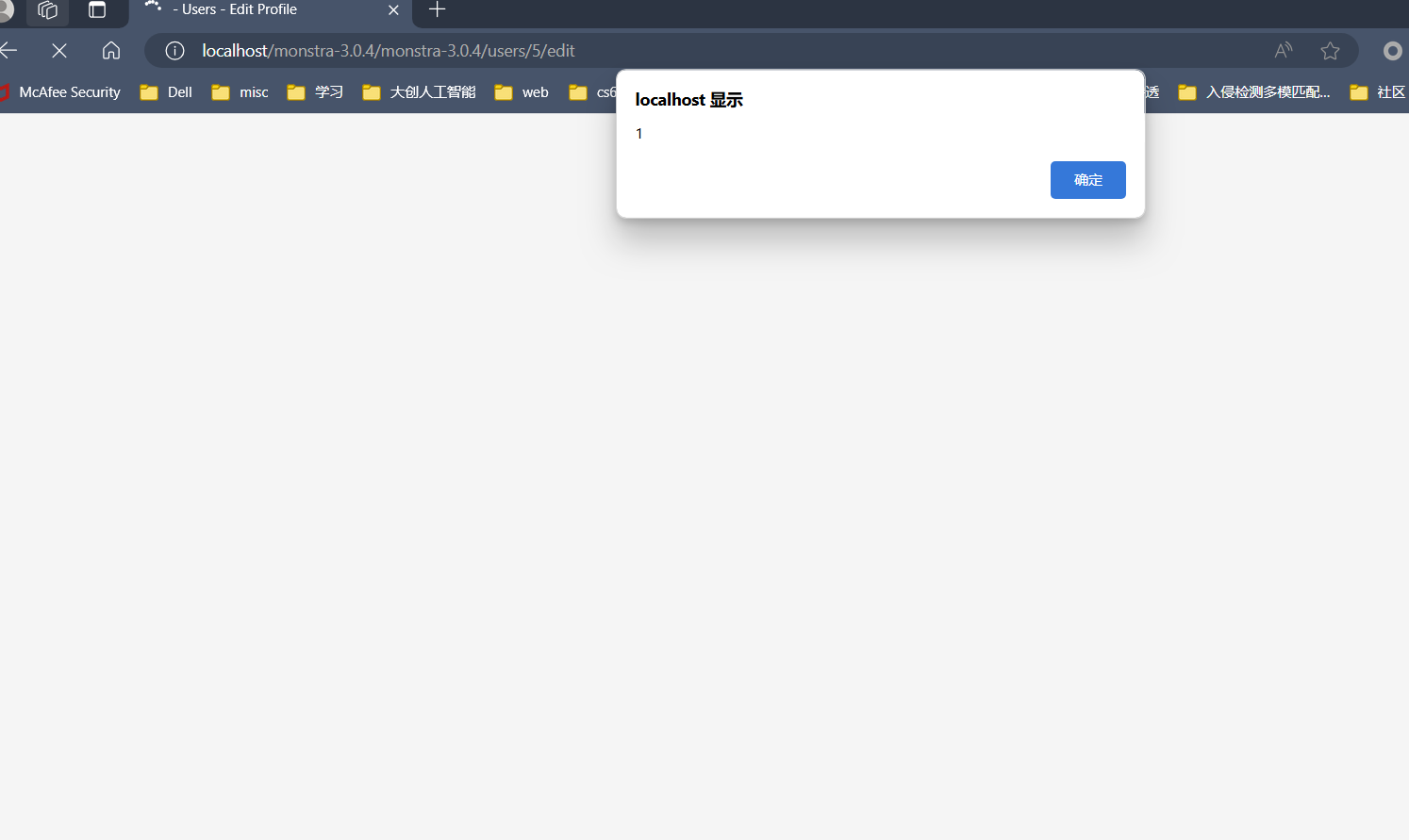
注册普通用户并登录，点击[Edit profile](http://localhost/monstra-3.0.4/monstra-3.0.4/users/5/edit)进入/monstra-3.0.4/users/5/edit

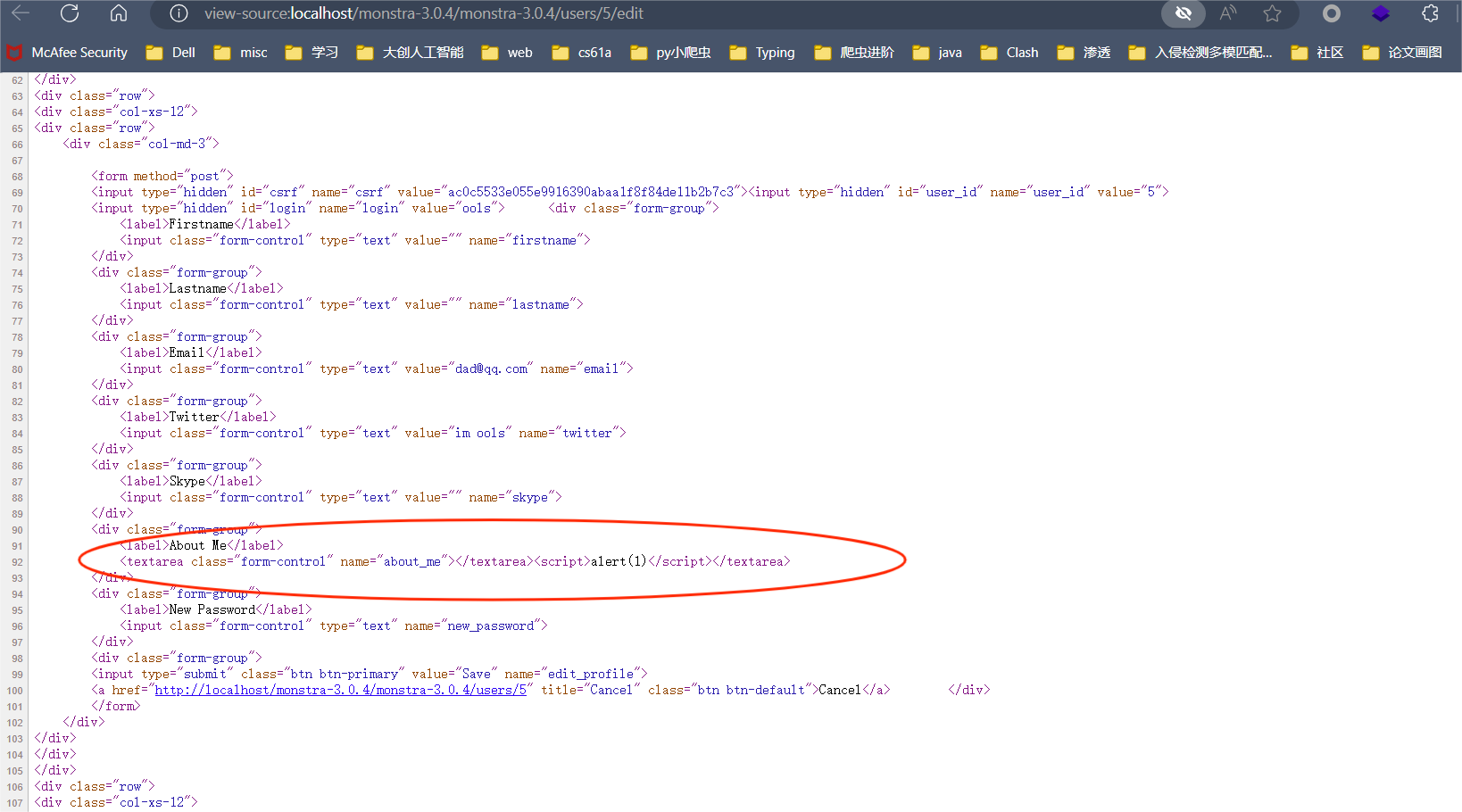


在About Me字段输入payload：</textarea><script>alert(1)</script>



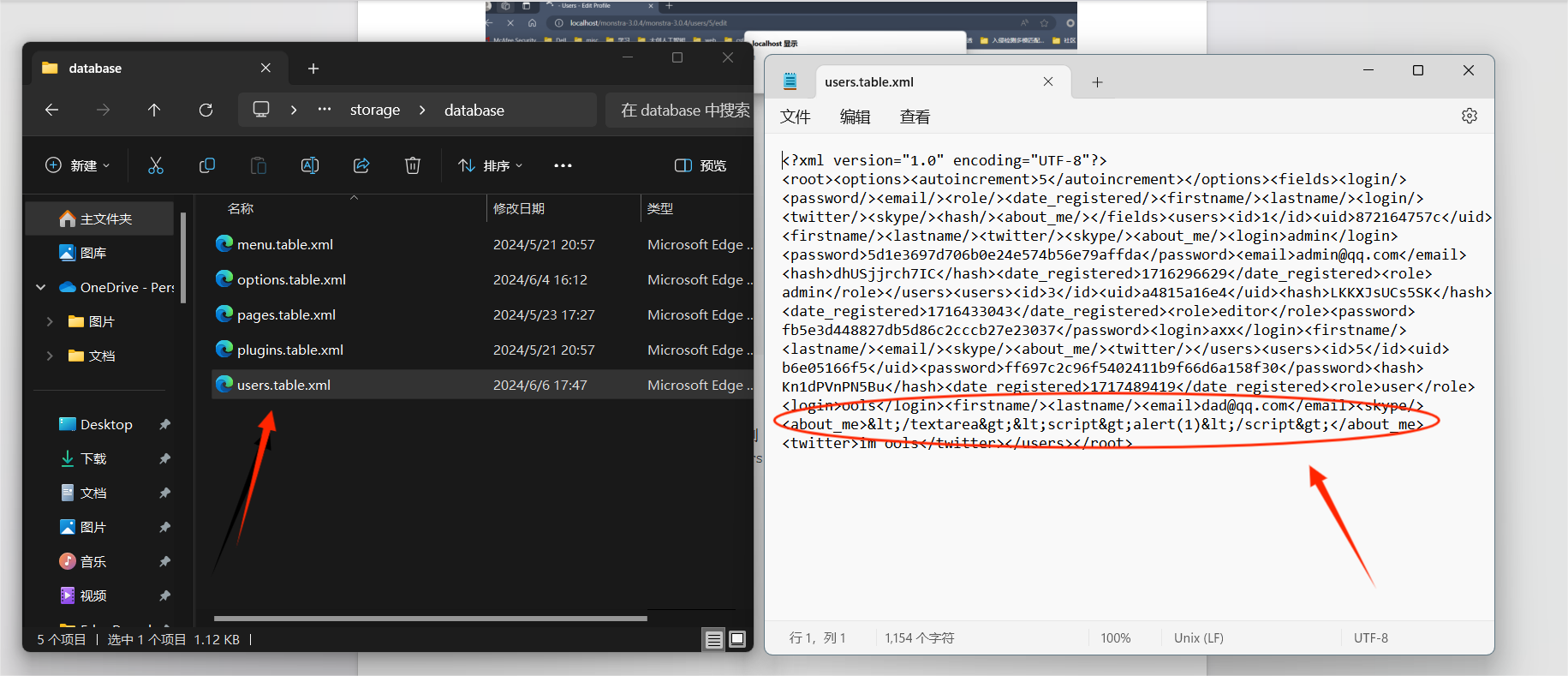
点击Save，退出后再次点击[Edit profile](http://localhost/monstra-3.0.4/monstra-3.0.4/users/5/edit)进入/monstra-3.0.4/users/5/edit





上述图片可以看到触发了XSS

该xss为存储型xss漏洞，刚刚输入的payload会存储在本地数据库\monstra-3.0.4\storage\database\users.table.xml中，以下是数据库中对应的内容



**漏洞成因：**

对于用户输入的数据</textarea><script>alert(1)</script>，

虽然经过htmlspecialchars()函数转义特殊字符”<”,”>”为&lt; &gt;后

再存储 &lt;/textarea&gt;&lt;script&gt;alert(1)&lt;/script&gt;

但是当网页再次通过以下方式读取时：

$xml = file\_get\_contents($file); $data = simplexml\_load\_string($xml);

$data->xpath()

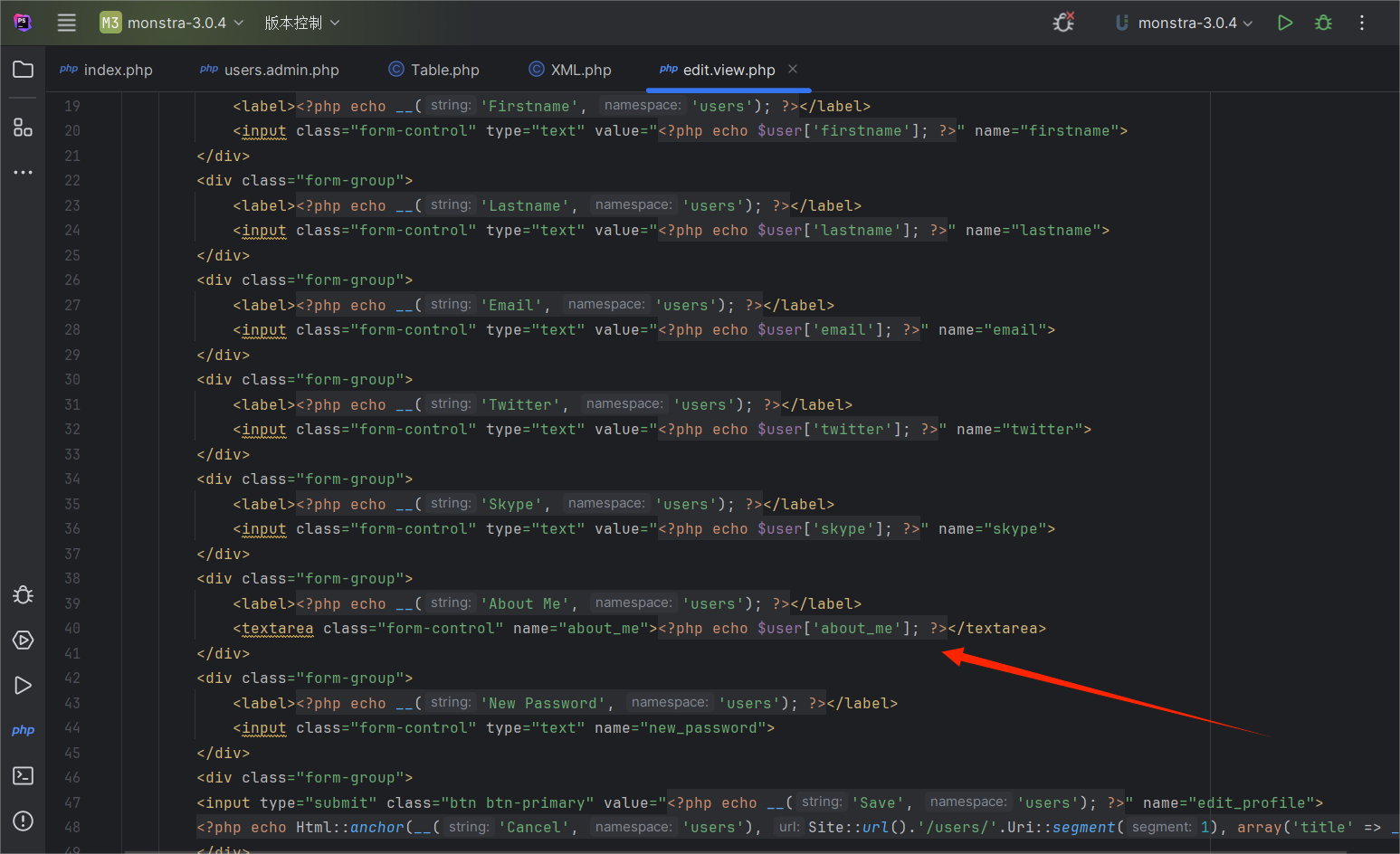
会将数据库中原本已经成功转义在存储的特殊字符重新变回原本的”<”和”>”

从而导致xss漏洞产生

**下面是源码调试过程：**

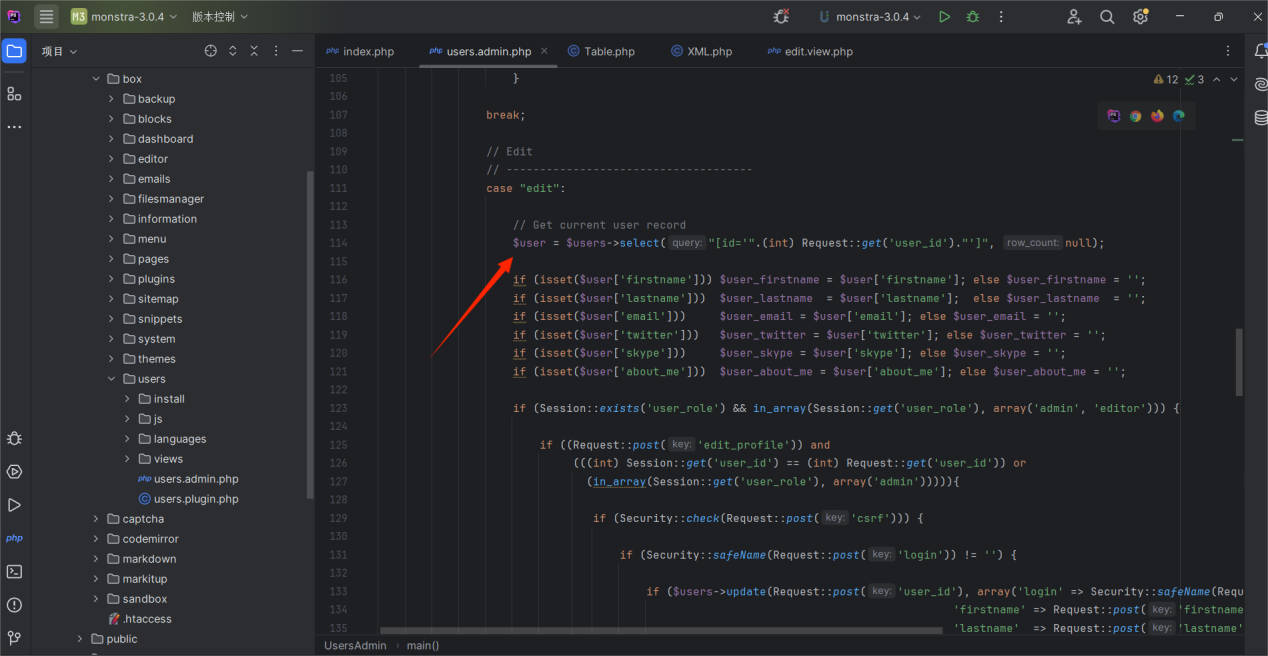
Xss触发路径为/monstra-3.0.4/users/5/edit，

该路径对应的php文件：monstra-3.0.4/plugins/box/users/views/frontend/edit.view.php



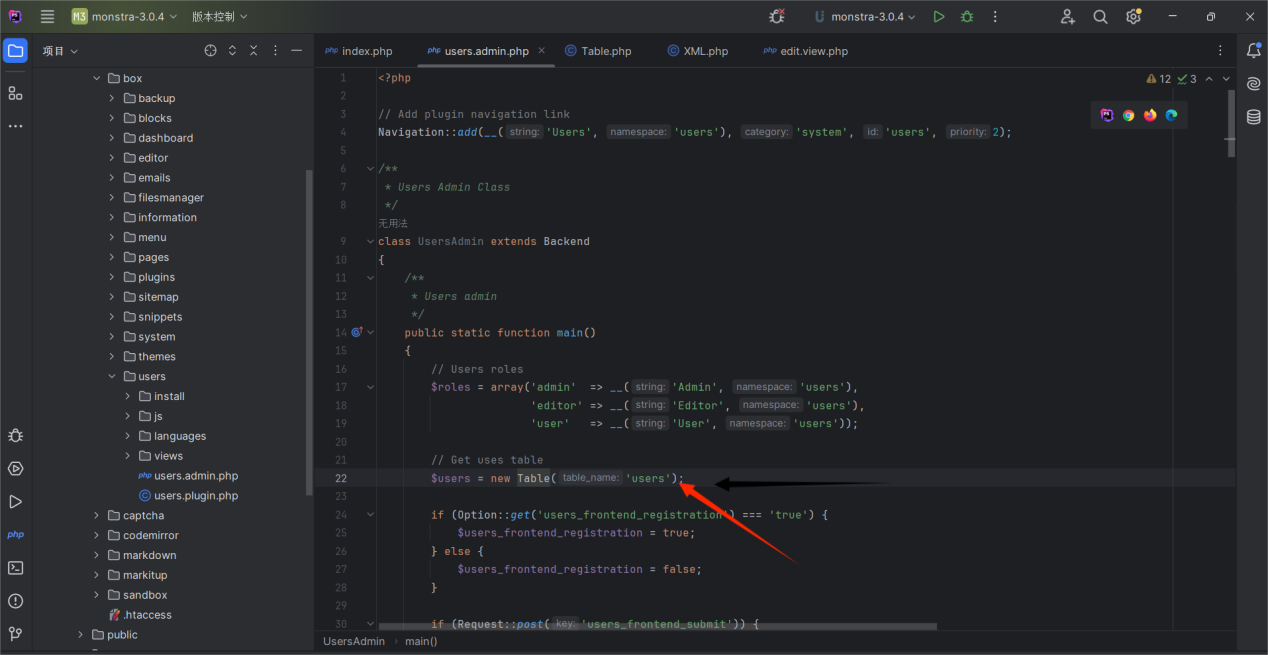
edit.view.php文件第40行，调用echo方法直接输出$user数组变量中的’about\_me’对应的value值；

而Array类型变量$user在edit.view.php的父文件users.admin.php中第114行定义，如下图



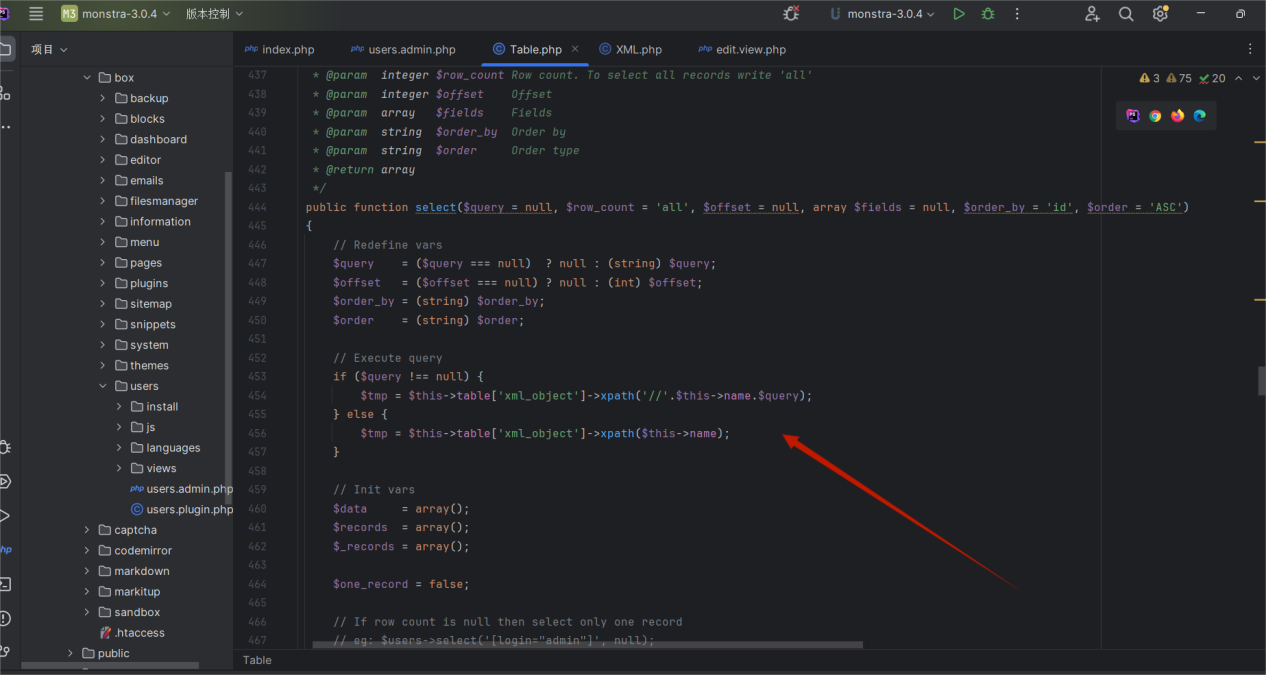
这一行代码中$users变量通过select方法，查询数据库中user\_id对于的数据并通过Array的形式返回给先前提到的$user变量。

$user变量在users.admin.php中第22行定义，是一个Table类

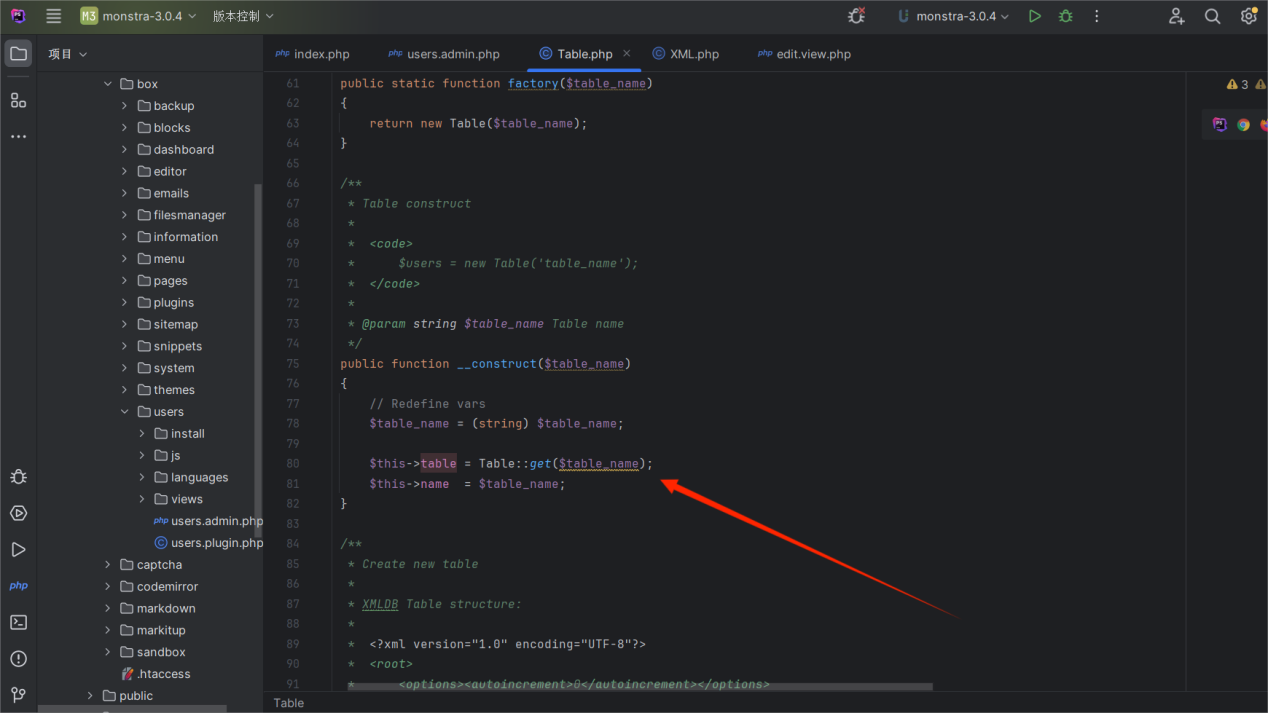


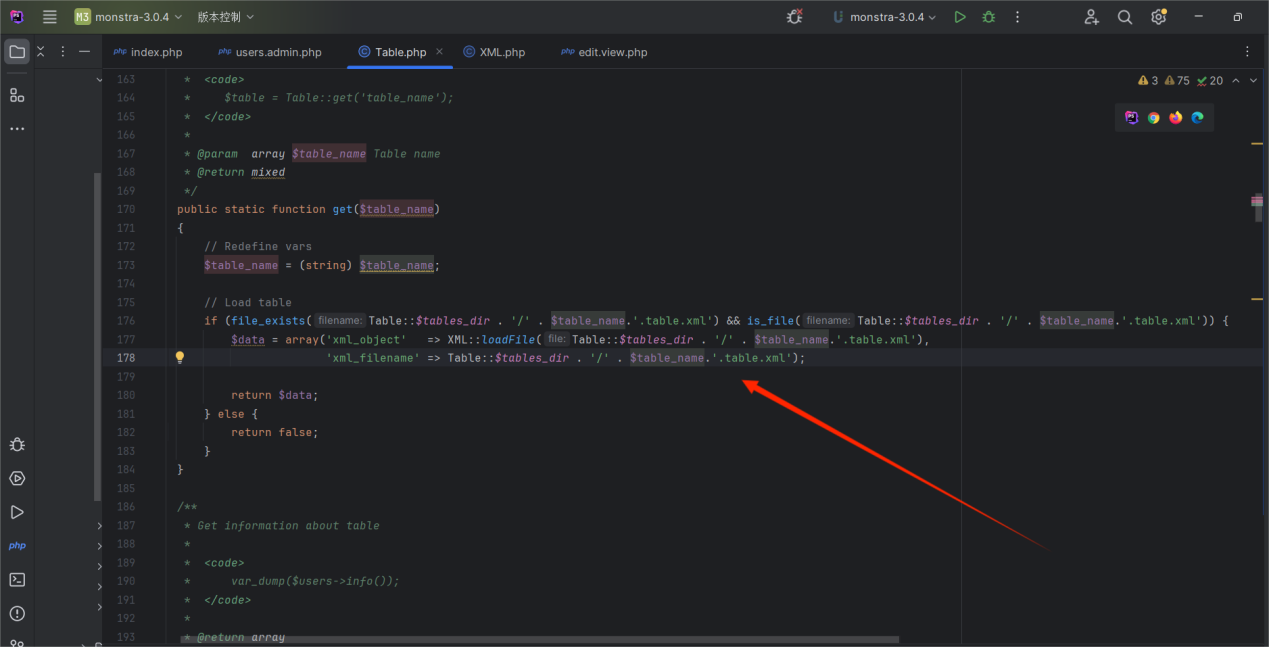
Table类在monstra-3.0.4/engine/Xmldb/Table.php文件中被定义

在Table.php文件中第444行定义select方法，其中在454行调用xpath方法来对xml类型的数据库对象进行数据查询，并接受返回值



这个xml\_object是Table类实例化时调用\_\_constract()方法给实例的内置Array类型变量table所绑定的key，它在创建实例时赋初值，是SimpleXMLElement类型对象。对应函数参考下图：



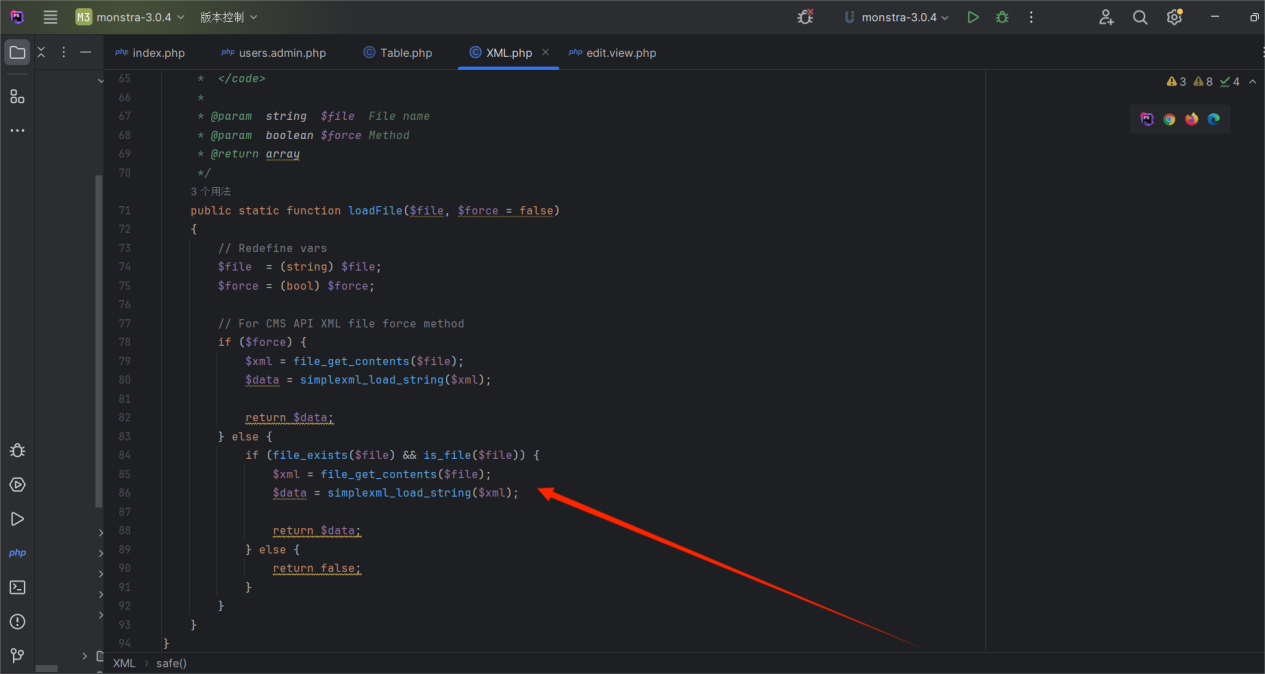


可以发现其中的XML::loadFile方法就是返回’xml\_object’对应的value取值

而在XML.php文件中的loadFile方法，最关键的两条函数便是

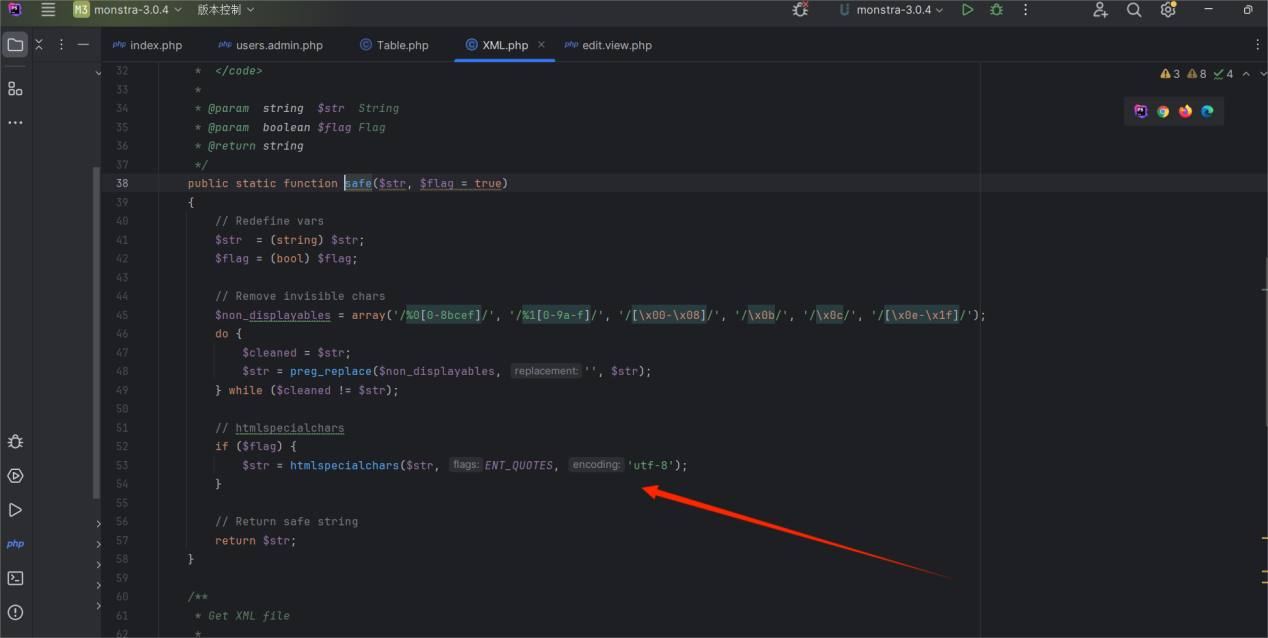
$xml = file\_get\_contents($file); $data = simplexml\_load\_string($xml);

最终return $data就是返回了SimpleXMLElement类型对象



补充payload存储时经过：

Payload在存储时会经过如下函数，被htmlspecialchar()转义特殊字符



**总结：**

**当用户输入payload，存储进数据库后，再次访问**/monstra-3.0.4/users/5/edit**时，会因为执行echo $user[‘about\_me’]实现xss，这个$user的值是通过Table类的select方法：识别user\_id拼接入xpath定位方法，查询对应数据库中的值并返回，生成SimpleXMLElement类型对象，再把SimpleXMLElement对象转换为对应Array类型对象然后retutn，最终赋值给$user[‘about\_me’]变量。虽然在用户写入payload时，会通过htmlspecialchars()函数转义特殊字符，当在读取数据时——“生成SimpleXMLElement类型对象”这一步会导致先前被转义字符变回原本字符形式，最终导致存储型xss漏洞。**