# CUMTCTF2017入门赛-Aegis writeup



weixin\_30399821 ① 于 2017-04-11 21:39:00 发布 O 405 论 收藏 文章标签: php python ux

原文链接: <u>http://www.cnblogs.com/yuriufo/p/7360847.html</u> 版权

# **Basic**

签到题 10

根据题目,百度得知,使用本地hosts文件进行域名解析。(虽然用hosts翻过墙,但并不知道原来是这么一回事)

#### 202.119.201.199 <sup>tp://</sup>welcome.bxsteam.xyz

WIN10 hosts目录: C:\Windows\System32\drivers\etc

添加如上字段即可。再刷新DNS即可。

C:\Users\Aegis>ipconfig/flushdns

\indows IP 配置

已成访刷新°™S 解析缓存/yuriufo

#### encode 10

题目如图:

CUMTCTF=~[];CUMTCTF={:++CUMTCTF,\$\$\$\$:(![]+"")[CUMTCTF],\$:++CUMTCTF,\$_
\$_:(![]+"")[CUMTCTF], \$_:++CUMTCTF,\$_\$\$:({}+"")[CUMTCTF],\$\$_\$:(CUMTCTF[CUM
TCTF]+"")[CUMTCTF],_\$\$:++CUMTCTF,\$\$\$_:(!""+"")[CUMTCTF],\$:++CUMTCTF,\$_\$:
++CUMTCTF,\$\$_:({}+"")[CUMTCTF],\$\$_:++CUMTCTF,\$\$\$:++CUMTCTF,\$_:++CUMTCTF
<pre>,\$\$:++CUMTCTF};CUMTCTF.\$_=(CUMTCTF.\$_=CUMTCTF+"")[CUMTCTF.\$_\$]+(CUMTCTF.</pre>
_\$=CUMTCTF.\$_[CUMTCTF\$])+(CUMTCTF.\$\$=(CUMTCTF.\$+"")[CUMTCTF\$])+((!CU
MTCTF)+"")[CUMTCTF\$\$]+(CUMTCTFCUMTCTF.\$_[CUMTCTF.\$\$_])+(CUMTCTF.\$=(!
""+"")[CUMTCTF\$])+(CUMTCTF=(!""+"")[CUMTCTF.\$_])+CUMTCTF.\$_[CUMTCTF.
<pre>\$_\$]+CUMTCTF+CUMTCTF\$+CUMTCTF.\$;CUMTCTF.\$\$=CUMTCTF.\$+(!""+"")[CUMTCTF</pre>
\$\$]+CUMTCTF+CUMTCTF.\$+CUMTCTF.\$\$;CUMTCTF.\$=(CUMTCTF)[CU
MTCTF.\$_][CUMTCTF.\$_];CUMTCTF.\$(CUMTCTF.\$(CUMTCTF.\$\$+"\""+CUMTCTF.\$_\$_+(![
]+"")[CUMTCTF\$_]+CUMTCTF.\$\$\$_+"\\"+CUMTCTF\$+CUMTCTF.\$\$_+CUMTCTF.\$_+C
UMTCTF+"(\\\\""+CUMTCTF.\$\$\$\$+(![]+"")[CUMTCTF\$_]+CUMTCTF.\$_\$_\$_+"\\"+CUM
TCTF\$+CUMTCTF.\$+CUMTCTF.\$\$\$+"{\\"+CUMTCTF\$+CUMTCTF.\$_\$+CUMTCTF.\$_
+"\\"+CUMTCTF\$+CUMTCTF.\$_\$+CUMTCTF\$_+"_"+CUMTCTF\$\$+"\\"+CUMTCTF\$
+CUMTCTF. \$+CUMTCTF.\$\$ +CUMTCTF.\$\$ +CUMTCTF. +CUMTCTF.\$\$ \$+CUMTCTF. \$\$
+"_\\"+CUMTCTF\$+CUMTCTF.\$_\$+CUMTCTF\$+CUMTCTF.\$_\$+"_"+CUMTCTF.\$\$\$\$+CU
MTCTF+"\\"+CUMTCTF\$+CUMTCTF.\$_\$+CUMTCTF/\$\$_(+2)\\\\"\\"+CUMTCTF.3\$1_+CUM
TCTF. +")"+"\"")())();

分析只可能是jsp混淆或者jjencode/aaencode加密(不确定)。 使用chrome/firefox自带的控制台即可自动解密。



# 滑稽 20 下载并打开ppt文件。发现ppt只有一页,猜想flag被隐藏在图片背后。



# check the ppt!

但是ppt无法编辑,尝试启动编辑。发现有密码: 由于ppt格式类似于zip压缩包。更改文件后缀名,得到huaji.zip 在以上路径找到3张图片。分析知image3.png有问题。之后启动16进制文本编辑器。 <!--pAsSW0Rd:Y3VtdDIwMTdjdGY=->

由此得到密码,但是密码经过Base64加密(有2个等号的特征)。之后解密得真正的密码。 Input your Decoding string:

Y3VtdDIwMTdjdGY=	^
	$\vee$
Base 64 Decode string :	
cumt2017ctf	$\wedge$
http://blog.csdn.net/vuriuf	

现在ppt可以编辑了,移动最上层的图片。可以发现flag。

# You found it !

flag{<u>hua\_tian\_xia\_zhi\_da\_j</u>i}

#### Fast 30

进入网页之后发现要在两秒内提交key。 fast × ● CUMTCTF ④ ③ web.bxsteam.xyz/fast/ 窗 百度一下,你就知道 → WEB ④ http://localhost/test... ⑦

#### Post what you find in 2s with param key

http://blog.csdn.net/yuriufo

#### 随后在消息头中找到flag,尝试提交后并不是答案。

消息头	Cookie	参数	响应
请求网址:http:// 请求方法:GET 远程地址:202.11 状态码: ● 200 ( 版本: HTTP/1.1	/web.bxsteam.x 9.201.199:80 OK	yz/fast/	
▼ 过滤消息头			
▼ 响应头 (0.354	KB)		
Cache-Contro	ol: "no-store, no	o-cache, must-r	evalidate"
Connection:	'keep-alive"		
Content-Enco	oding: "gzip"		
Content-Typ	e: "text/html; ch	narset=UTF-8"	
Date: "Sun, 0	9 Apr 2017 10:0	2:33 GMT*	
Expires: "Thu	, 19 Nov 1981 0	8:52:00 GMT*	
Flag: *149173	2153"		
Pragma: "no-	cache"		
Server: "ngin:	x*		
Transfer-Enc	oding: "chunke	d"	
Vary: "Accept	-Encoding"		
X-Powered-E	PHP/7.0.75	. csdn. net	/yuriufo
	1/0)		

#### 又发现因为要在2s内提交,故编写python脚本如下,得到flag。



111

Þ.

< [

**LSB 30** 



由于题目提示了隐写的类型为LSB。故使用Stegsolve工具帮助分析。 由典型文件头,保存以上数据为png文件。 由此得到一个二维码。



扫描该二维码即可得到flag。

## 视而不见 50

由题目知是一张3D图片。Flag隐藏在图片中。开始尝试用photoshop分析,但是失败了。之后用StegSolve打开,选择Stereogram Solver。

StegSolve 1.3 by Caesum



#### 最终找到flag。



# 鲨鱼的套路 70

#### 用wireshark打开后,根据hint查看TCP流。

10.000000	192.168.2.55	74.125.204.10	32 TCP	66 48866+443 [SYN] Seq=0 W1n=8192 Len=0 MSS=
2 0.090262	192.168.2.50	255.255.255.2	UDP	215 57856+7437 Len=173
3 3.162208	192.168.2.50	255.255.255.2	255 UDP	215 57856+7437 Len=173
4 5.152726	140.206.78.8	192.168.2.55	TCP	56 80+44963 [ACK] Seq=1 Ack=1 Win=501 Len=0
5 5.152901	192.168.2.55	140.206.78.8	TCP	54 [TCP ACKed unseen segment] 44963+80 [ACK]
6 5.781489	192.168.2.55	192.168.2.57	TCP	66 48069+8000 [SYN] Seq=0 Win=8192 Len=0 MSS
7 5.796462	192.168.2.57	192.168.2.55	TCP	68 8000+48069 [SYN, ACK] Seq=0 Ack=1 Win=292
8 5.796599	192.168.2.55	192.168.2.	EC/RIMEC ORAN	[00 [ACK] Seq=1 Ack=1 Win=16384 Le
9 6.003382	192.168.2.55	74.125.204	104C/404404C 2018(M)	CtdeD ransmission] 48066+443 [SYN] Seq-
10 6.132239	192.168.2.50	255.255.25	10 00 /00 /00 00 01/00 01/00 04	CrideT 37 Len=173
11 7.972594	192.168.2.55	74.125.204	REAL PROPERTY AND A REAL P	Cridashidaa B [SYN] Seq=0 Win=8192 Len=0 MSS=
12 8.058189	192.168.2.55	192.168.2.	049218	CritaAltaC 00 [PSH, ACK] Seq=1 Ack=1 Win=163
13 8.072516	192.168.2.57	192.168.2.	73 NEW 2 Perce	69 [ACK] Seq=1 Ack=2 Win=29312 Le
14 8.203677	192.168.2.55	192.168.2.	编辑解析的名称	00 [PSH, ACK] Seq=2 Ack=1 Win=163
15 8.220420	192.168.2.57	192.168.2.	作为过滤器应用	69 [ACK] Seq=1 Ack=3 Win=29312 Le
16.8.399261 192.168.2.55 192.168.2.		192.168.2.	准备过滤器	, PA [PSH. ACK] Sen=3 Ack=1 Win=163
ame 8: 54 bytes o hernet II, Src: P ternet Protocol V	on wire (432 bits), 5 HonHaiPr_8c:e0:73 (c0 Mersion 4, Src: 192.1	4 bytes capti :38:96:8c:e0 68.2.55, Dst	对话过滤器 对话看色 SCTP	, 9:c4:55:bc:e1)
ansmission Contro	ol Protocol, Src Port	: 48069, Dst	追踪流	• TCP 选
			SEM	, UDP I瓶
44 39 c4 55 bc 00 28 51 8d 40 02 39 bb c5 1f	e1 c0 38 96 8c e0 3 00 40 06 63 82 c0 40 f0 36 25 08 1d	73 08 00 45 0 88 02 37 c0 a 74 16 11 50 1	协议普选项 解码为(A) 在新窗口显示分组(W)	SSL 30. HTTP 30.

#### 找到如下信息。

hello hehe i will give you flag~ o but you must tell me te.he anhhaoao~ wwyaohuinongcun ok i will give you flag~ just reviceg.receive it UESDBBQAAQAZIANeFferHickKomAABSAAAACAAAAaGludF9zb211IF90aGluZ19pc19mYwtllnR4dEvLSUyvTs51zUuPL87IjC9)zI/PKQUyU/NqAVBLAQI/ ABQACQAIANeFferHickKomAABSAAAACAAAAaGludF9zb211IF90aGluZ19pc19mYwtllnR4dEvLSUyvTs51zUuPL87IjC9)zI/PKQUyU/NqAVBLAQI/ ABQACQAIANeFferHickKomAABSAAAACACQAAAAAAAAAGAUdF9zb211IF90aGluZ19pc19mYwtllnR4dEvLSUyvTs51zUuPL87IjC9)zI/PKQUyU/NqAVBLAQI/ ABQACQAIANeFferHickKomAABSAAAACACQAAAAAAAGAUdF9zb211IF90aGluZ19pc19mYwtllnR4dEvLSUyvTs51zUuPL87IjC9)zI/PKQUyU/NqAVBLAQI/ ABQACQAIANeFferHickKomAABSAAAACACQAAAAAAAAAAAAAAAAAAAA= ~~~~~~~~~~~~~~~~~	【Wireshark - 追踪 TCP 流 (tcp.stream eq 2) - flag	-		
behe i will give you flag~ o but you must tell me te.he anhhaoao~ wwyaohuinongcun ok i will give you flag~ just reviceg.receive it UESDBBQAAQAZIMMeFferHickGomAABASAAAACAAAAaGludF9zb211IF90aGluZ19pc19mYwtllnR4dEvLSUyvTs51zUuPL87IjC9)zI/PKQUyU/NqAVBLAQI/ ABQACQAIXMeFferHickGomAABSAAAACAAAAaGludF9zb211IF90aGluZ19pc19mYwtllnR4dEvLSUyvTs51zUuPL87IjC9)zI/PKQUyU/NqAVBLAQI/ ABQACQAIXMeFferHickGomAABSAAAACACAQAAAAAAAIAAAAAAAAAAAAAAAAAAAAAA	hello			
<pre>i will give you flag- o but you must tell me te.he anhhaoao- woyaohuinongcun ok i will give you flag- just reviceg.receive it UssBBBQAAAgMaindefferHigkGkwaAABAAAAAAAAGIudF9zb211F90aGluZ10pc10mYwtllnRAdEvLSUyvTsSIzUuPL87IjC93zI/PKQUyU/NqAVBLAQI/ ABQWCQAIANefferHigkGkwaAABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA</pre>	hehe			
0 but you must tell me te.he anhhaoao- woyaohuinongcun ok i will give you flag- just reviceg.receive it UESDBBQMAAgIAMeffErHFL6KGwAAABSAAAACAAAAaGludF9zb211IF90aGlu219pc19mYwtlLnR4dEvLSUyvTS5IzUuPL87IjC9JzI/PKQUyU/NqAVBLAQI/ ABQACQAIXMeffErHFL6KGwAAABsAAAACAQAAAAGludF9zb211IF90aGlu219pc19mYwtlLnR4dEvLSUyvTS5IzUuPL87IjC9JzI/PKQUyU/NqAVBLAQI/ ABQACQAIXMeffErHFL6KGwAAABsAAAACAQAAAAGludF9zb211IF90aGlu219pc19mYwtlLnR4dEvLSUyvTs5IzUuPL87IjC9JzI/PKQUyU/NqAVBLAQI/ ABQACQAIXMeffErHFL6KGwAAABsAAAACAQAAAAGludF9zb211IF90aGlu219pc19mYwtlLnR4dEvLSUyvTs5IzUuPL87IjC9JzI/PKQUyU/NqAVBLAQI/ ABQACQAIXMeffErHFL6KGwAAABsAAAACAQAAAAAAAAAAGAAAAAAAAAAAAAAA mogGhDeAnl6fSAaEHACeXp9IBUESFBBAAAAABAAEAbgAAAFUAAAAAAA== > bye=	i will give you flag~			
but you must tell me te.he anhhaoao- woyaobuinongcun ok i will give you flag- just reviceg.receive it UESBBQAAQAZIMWEFferHickGwaAABSAAAACAAAAaGludF9zb211IF90aGluZ19pc19mYwtllnR4dEvL5UyvTs5IzUuPL87IjC9JzI/PKQUyU/NqAVBLAQI/ ABQACQAIXHEFfErHIcGKGwAAABSAAAACACQAAAAAAAIAAAAAAAAAAAAAAAAAAAAAAA	0			
woyabulinongcun woyabulinongcun woi will give you flag- just reviceg.receive it UESDBQAAQAIIINNEFfErHELSKGWAAABAAAAAAAAAGUUdF9zb211F90aGluZ19pc19mYNtllnRAdEvLSUyvTs5IzUUPL87TjC9JzI/PKQUyU/NqAVBLAQI/ ABQACQAIINNEFFErHELSKGWAAABAAAAAAAAAAAAGUUdF9zb211F90aGluZ19pc19mYNtllnRAdEvLSUyvTs5IzUUPL87TjC9JzI/PKQUyU/NqAVBLAQI/ ABQACQAIINNEFFErHELSKGWAAABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA HMGGDDeAnl6fSAaEH4CeXp9IBUEsFBgAAAAABAAEAbgAAAFUAAAAAA= ~ bye- 0	but you must tell me te.he anhhaoao~			
ok i will give you flag~ just reviceg.receive it USSBBQAAAgIANeffErHFL6KGGAAABSAAAACAAAAaGludF9zb21lTF90aGluZ19pc19mYWtlLnR4dEvL5UyvT55IzUuPL87IjC9JzI/PKQUyU/NQAVBLAQI/ ABQACQAIXMeffErHFL6KGGAAABSAAAACACQAAAAAAAGludF9zb21lTF90aGluZ19pc19mYWtlLnR4dEvL5UyvT55IzUuPL87IjC9JzI/PKQUyU/NQAVBLAQI/ ABQACQAIXMeffErHFL6KGGAAABSAAAACACQAAAAAAAAAAAAAAAAAAAAAAAAAAA	woyaohuinongcun			
just reviceg.receive it UESDBQAAAgAIstusefferteickiskanaasaAAAcAAAAGIudF9zb21lIF90aGIuZ19pc19mYWtllnRAdEvLSUyvTsSIzUuPL87IjC9JzI/PKQUyU/NQAVBLAQI/ ABQACQAIANEFfErHFL6KSmAABSAAAAcAAAAAGAAAAAAAAAAAAAAAAaaGaWS9X3NvbMUgX3RoaWSnX2lzXZzha2UudHh0CgAgAAAAAABABABgAu7Ux13 +m0gGNDeAnIdfSAaEH4CeXp9IBUEsFBgAAAAABAAEAbgAAAFUAAAAAA= ~ bye- o	ok i will give you flag~			
UESDBQAAAgIXHeFFErHFL6KGwAAABSAAAACAAAAAGIJudF92b211FP9aGIU219pc19mYktlLnRAdEvLSUyvTsSIZUUPL87IjC93zI/PKQUyv/NqAYBLAQI/ ABQACQAIXHeFFErHFL6KGwAAABSAAAACACQAAAAAAAAAAAAAAAAAAAAAAAAAAA	just reviceg.receive it			
ABQACQATANIEFFErNFL6KGwAAABsAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA	UESDBBQAAAgIANeFfErNFL6kGwAAABsAAAAcAAAAaGludF9zb21lIF90aGluZ19pc19mYWtllnR4dEvLSUyvTs5IzUuPL87IjC9JzI/PKQU	/U/NqAV	BLAQI/	
+n0gGhDeAn16fSAaEN4CeXp91BUESFBgAAAABAAEAbgAAAFUAAAAAA== ~ bye~ 0	ABQACQAIANeFFErNFL6kGwAAABsAAAAcACQAAAAAAAAAAAAAAAAAAAAAAABoawS0X3NvbWUgX3RoawSnX21zX2Zha2UudHh0CgAgAAAAAAABABg	u7Ux13		
∼ bye∽ o	+n8gGhDeAn16fSAaEN4CeXp9IBUEsFBgAAAAABAAEAbgAAAFUAAAAAAA==			
o bye~				
0	bye~			
	0			

#### http://blog.csdn.net/yuriufo

UEsDBBQAAAgIANeFfErNFL6kGwAAABsAAAAcAAAaGludF9zb21llF90aGluZ19pc19mYWtl LnR4dEvLSUyvTs5lzUuPL87ljC9Jzl/PKQUyU/NqAVBLAQI/ABQACQAIANeFfErNFL6kGwAAABsAAAAa ACQAAAAAAAAAAAAAAAAAABoaW50X3NvbWUgX3RoaW5nX2lzX2Zha2UudHh0CgAgAAAAAA BABgAu7Ux1J+n0gGhDeAnl6fSAaEN4CeXp9IBUEsFBgAAAAABAAEAbgAAAFUAAAAAAA= 分析知是base64加密,对其解码成十六进制得到。

68 69 6e 74 5f 73 6f 6d 65 20 5f 74 68 69 6e 67 5f 69 73 5f 66 61 6b 65 2e 74 78 74 0a 00 20 00 00 00 00 00 01 00 18 00 bb b5 31 d4 9f a7 d2 01 a1 0d e0 27 97 a7 d2 01 a1 0d e0 27 97 a7 d2 01 50 4b 05 06 00 00 00 01 00 01 00 6e 0. 0 00 00 55 00 00 00 00 00

由文件头知是明显的zip压缩包格式。保存为zip格式,尝试打开包。发现有密码。打开16进制文本编辑器。



其中发现存在伪加密,将光标处的9改成0保存,打开即得到flag。

1 217	大小	压缩后大小	类型
(上层目录)			
hint_some _thing_is_fake.txt	1 KB	1 KB	文本
hint come thing is take tyt.	104		
<pre>文件(F) 编辑(E) 格式(O) 查看(V) flag{cheng_shi_tao_lu_she</pre>	1C争本 帮助(H) en}		
文件(F) 编辑(E) 格式(O) 查看(V) flag{cheng_shi_tao_lu_she http://b	<del>IC事本</del> 帮助(H) en} log.csdn.i	net/yuriu	fo

#### easy crypto 70

打开文件后发现两个txt。

段全 📙 easy_crypto.zip\easy_c	rypto
+ 名称	大小 压缩
(上层目录)	
DS_Store	6 KB
flag.txt	1 KB
morse.txt tp://blog	.csdn.net <b>%</b> riufo

两个都打开后,发现morse.txt中明显是摩斯电码。随后解码如图。

结果: KEYISVIGENERECIPHER1010

Key is vigenerecipher意为维吉尼亚,即知道flag是维吉尼亚密码加密。

密钥: VIGENERECIPHER 密文框: vigenereencryptionhhh pg.csdn.net/yuriufo

解密便得到flag。

#### 学姐真美 120

开始并没有什么思路,之后放出hint:图片中存在另外一张png图片。随后打开16进制文本编辑器。 <sup>0001a390</sup> <sup>0001a390</sup> <sup>00001a390</sup> <sup>00001a300</sup> <sup>00001a300</sub> <sup>00001a300</sup> <sup>00001a300</sub> <sup>00001a300</sup> <sup>00001a300</sub> <sup>00001a300</sup> <sup>00001a300</sub> <sup>00001a3</sup></sup></sup></sup></sup></sup></sup></sup></sup></sup></sup></sup></sup></sup></sup></sup></sup></sup></sup></sup>

发现在图片最后,即0xffd9之后,出现了典型的png文件头格式(被修改了部分)。

00000020h: 93 00 00 00 09 70 48 59 73 00 00 0A 75 00 00 0A Pt. pHYsL u.U f 0

搜索资料得知(上图为正确的png格式文件头)。 由此将0xffd9之后的数据分离出来。修改部分文件头,得到一个png图片。

# Charles and the set of the set of

发现是一个不完整的图片,猜测是二维码。根据提示,再次对照png文件头,可能是认为修改了图片长宽导致显示不正确。

IHDR

文件头数据块IHDR(header chunk):它包含有PNG文件中存储的图像数据的基本信息,并要作为第一个数据块 出现在PNG数据流中,而且一个PNG数据流中只能有一个文件头数据块。

文件头数据块由13字节组成,它的格式如下表所示。

	域的名	3称	WP.	字节数					ÿ	明							
Widt	h		4	bytes	图像	宽度,	以像	素为单	位				net	/			
Heigl	ht		4	bytes	图像	高度,	以像	素为单	位	og.	csu			y y u	r i u	110	
再次查找资	{料,	手ž	动修	改了	て件	头。	0x	10开	头〉	<b></b>	和高	<u>,</u>	因为	是	二绡	<b>陷</b> ,	改成相同即可。
00000247	00	01	02	03	04	05	06	07	08	09	0a	Øb	0c	0d	0e	0f	
					100												

00000000	89	50	4e	47	Ød	0a	1a	0a	00	00	00	Ød	49	48	44	52
00000010	00	00	01	18	00	00	01	18	08	02	00	00	00	08	ec	7e
00000020	db	00	00	05	c7	49	44	41	54	78	9c	ed	dd	41	8e	e4
00000030	36	10	00	41	b7	b1	ff	ff	f2	fa	eØ	b3	0e	5c	67	b9

#### 扫码即可得pass。



111

Þ.

WEB

源代码	10
-----	----

查看	<b>f</b> 源代码最后	言一行,把	文字删除!	即是flag。			
R	○ 查看器	▶ 控制台	□ 调试器	{}样式编辑器	② 性能	□ 内存	三 网
+							
	<span style="co&lt;/td&gt;&lt;td&gt;lor:#ffe345">(</span>						
	<span style="co&lt;/td&gt;&lt;td&gt;lor:#ffe345">F</span>						
	<span style="co&lt;/td&gt;&lt;td&gt;lor:#ffe345">S</span>						
	<span style="co&lt;/td&gt;&lt;td&gt;lor:#ffe345">T</span>						
	<span style="co&lt;/td&gt;&lt;td&gt;lor:#ffe345">U</span>	<pre>k/span&gt;</pre>					
	<span style="co&lt;/td&gt;&lt;td&gt;lor:#ffe345">\</span>						
	<span style="co&lt;/td&gt;&lt;td&gt;lor:#ffe345"></span>						
	<span style="co&lt;/td&gt;&lt;td&gt;lor:#ffe345">)</span>						
	<pre><span style="co.&lt;/pre&gt;&lt;/td&gt;&lt;td&gt;lor:#ffe345">)</span></pre>						
	<span style="co&lt;/td&gt;&lt;td&gt;lor:#ffe345">2</span>						
	<span style="co&lt;/td&gt;&lt;td&gt;lor:#ffe345">#</span>						
	<span style="co&lt;/td&gt;&lt;td&gt;lor:#ffe345">B</span>						
	<span style="co&lt;/td&gt;&lt;td&gt;lor:#ffe345">C</span>						
	<span style="co&lt;/td&gt;&lt;td&gt;lor:#ffe345">D</span>						
	<span style="co&lt;/td&gt;&lt;td&gt;lor:#ffe345">8</span>						
	<span style="co&lt;/td&gt;&lt;td&gt;lor:#ffe345">F</span>						
	<span style="co&lt;/td&gt;&lt;td&gt;lor:#ffe345">0</span>	i					
	<span style="co&lt;/td&gt;&lt;td&gt;lor:#ffe345"></span>						
	一些节点已隐藏。	再显示一个节	5点;显示全部。	9999 个节点			
<	/body>						
<td>tml&gt;</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>	tml>						
<1-	-该f配1合a你g演(出)	/的o我u演f视i而	in不d见m别e逼2一	3个3最3爱3你3的3人	3即3兴3表3演}	not /wiri	ufo_
h	tml > body >	pre > sp	an	noop.//bi	08.0501.	net/ yur	uio

## 请多多"包含"! 100

打开后发现file=page,明显的文件包含题。 健 CUMTCTF × 百度一下,你就知道 X () web100.bxsteam.xyz/index.php?file=page / yuriufo 查看想要看的代码file=php://filter/read=convert.base64-encode/resource=index.php CUMTCTF × 百度一下,你就知道  $\times$  http://web100...urce=./index  $\times$ ♦ ③ web100.bxsteam.xyz/index.php?file=php://filter/read=convert.base64-encode/resource 📸 百度一下 , 你就知道 📒 WEB ✓ ■ ♦ SQL\* XSS\* Encryption\* Encoding\* Other\* INT Load URL //flag{e45y IncLud3} //check other files to get more flag! get the shell! Split URL ?> Execute

Enable Post data
 Enable Referrer

PCEtLXVwbG9hZC5waHAtLT4KPD9waHAKaWYoaXNzZXQoJF9SRVFVRVNUWydma\

http://blog.csdn.net/yuriufo

注解:

1.php://filter/可用于处理打开的数据流,起到过滤作用。如果源文件为.php则很有可能在前台显示不出来。 2.此时我们采用的方法是,先让文件转化为base64格式(convert.base64-encode)然后再输出,这样不论是什 么格式的文件都可以在前台输出。 输入后将Base64解码得到flag.

#### 世界上最好的语言 100

#### php代码审计

要使输入的hash与key的md5解码相同,且存在unserialize反序列化,根据类构造实例。

( i localhost/test.php	)
📓 百度一下 , 你就知道 🔒 WE	В
O:4:"Cumt":1:{s:3:"key";	s:8:"flag.php";}
E:\wamp\www\test.php	- Notepad++
文件(E)编辑(E)搜索(S)视	图(V)编码(N)语言(L)设置(I)工具(Q)宏(M)运行(B)插件(P) 【 🌗 🌔 💭 🛃 🛗 🆖 🔍 🔍 ဩ 强 🔜 🗐 🎼 📿 圆 🌡
🔚 test. php🛛	
1 ⊟ php<br 2 //flag is in f 3 Class Cumt	lag.php
5 public \$ke	y='flag.php';
6 -} 7 echo seria	lize(new Cumt);
8 - ?>	http://blog.csdn.net/yuriufo
但因为要绕过 wakeu	<b>Ø</b> ().所以百度后可知将
O:4:"Cumt":1:{s:3:"k	<pre>(ey";s:8:"flag.php";}</pre>
改成:	
O:4:"Cumt":2:{s:3:"k	<pre>(ey";s:8:"flag.php";}</pre>
并与hash值一起提交,即	J可看到flag。



http://blog.csdn.net/yuriufo

MO	B	L	Ε

•

#### \_\_\_\_

# Gift 10

由于是安卓题,使用androidkiller打开apk文件。但是并不懂smali语言(其实java也不懂)。



无奈下直接利用工具反汇编apk。在包中找到明码flag。

## Hide&Seek 100

开始并没有什么头绪,直到给出hint,运用了最基础的图片隐写。便直接分析apk包中的图片。

111



/blologo.pngn.net/logo\_pngfo

发现这样两个图片,为logo,纯粹猜测logo\_.png有问题(因为打不开)。 用16进制文本编辑器分析,在最后一行出现异常:

#### 釜瀨镉{OHYOVF1NDM3}

尝试以flag{0HY0VFINDM3}进行提交,成功。。。

#### **Reverse**

•

#### Easy\_CrackMe 50

把程序下载下后,从初次运行可以看出是一个Win32 GUI平台。 .text:00401000 ; int \_\_stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCndLine, int nShowCnd) .text:00401000 \_VinMain016 proc near ; CODE XREF: start+C91p .text:08401000 .text:00401000 hInstance .text:00401000 hPrevInstance .text:00401000 lpCmdLine .text:00401000 nShowCmd dword ptr
dword ptr
dword ptr 8Ch - dword ptr 10h .text:00401000 .text:00401000 .text:00401004 .text:00401004 eax, [esp+hInstance] ; dwInitParan push offset DialogFunc ; 1pDialogFunc 0 : hVndParent push push text:08401008 .text:00401000 .text:0040100F ; 1pTenplateNam ; hInstance 65h push eax ds:DialogBoxParanA .text:00401010 call .text:00401016 xor eax, eax .text:00401018 .text:00401018 \_WinMain@16 retn endp

拖进IDA中查看,主函数(入口地址)如图:

左侧可以看到很多函数,但看到主函数只调用了DialogBoxParamA一个函数(参数如下)

DialogBoxParamA(创建窗口){

hlnstance是当前应用程序的实例句柄。

lpTemplateName是对话框的资源模板。

hWndParent是父窗口的句柄。

lpDialogFunc是对话框的消息处理函数。

dwInitParam是初始化参数。

可知,只有lpDialogFunc是消息处理函数,即输入对话框中的值(消息)只和该函数有关。

asy	CrackMe	_		×
			?1	,
	http://blog.cs	dn.net/	/yuri	ufo

跟进DialogFunc,该函数只调用了sub\_401080(地址为0x401080处的函数),再次跟进。

第一部分:

}

首先调用了GetDigItemTextA的API函数(进行输入),注意传参顺序。Mirosoft采用stdcall即标准调用规则。参 数由右向左依次push,显然lpString为输入的值(形参),即eax。

```
edi, [esp+6Ch+hDlg]
mnu
        eax, [esp+6Ch+String]
lea
                         ; 1pString
push
        eax
push
        3E8h
                         ; nIDD1gItem
push
        edi
                         ; hDlq
call
        ds:GetDlgItemTextA
        [esp+68h+var_63],/yariufo
CMD
        short loc_401135
jnz
```

而在push eax 之前 由lea指令取String变量的地址,所以String为输入值(实参),

再看变量的定义,可以发现有4个变量,图中表示距离某个基址(ebp)的偏移量,所以:String,var\_63,var\_62, 分别为1,1,2个字节。

String= byte ptr -64h var\_63= byte ptr -63h var\_62= byte ptr -62h var\_60= byte ptr -60h

之后便是这句: cmp指令,比较前后两个值是否相等(简单理解就是var\_63与a是否相等),jnz表示若不相等则 跳转道loc 4001135处

cmp ://l[esp+68h+var\_63], 'a' inz ://bhort loc 401135

不用说,看字符串就直到, incorrect, 密码不对。所以var\_63=a

```
loc 401135:
                         ; uType
push
        10h
        offset Caption ; "EasyCrackMe"
push
        offset aIncorrectPassw ; "Incorrect Password"
push
                         ; hWnd
push
        edi
call
        ds:MessageBoxA
        edi
DOD
add
        esp, 64h
retn
sub_401080 endp
第二部分:
```

push 2 ; size\_t lea ecx, [esp+6Ch+var\_62] offset a5y ; "59" push push ecx ; char \* call strncmp esp, OCh hhs. gax, gax, net/vuriufo short loc\_401135 bloeaxeseax.net test jnz

显然是push3个参数后调用strcmp函数进行对比,比较长度2字节,这与上面推导的var\_62有2字节相同,所以 var 62=5y(不然又会显示incorrect)。

```
第三部分:
|push ebx
|push esi
mov esi, offset_aR3versing;"|R3versing"
|lea eax, [esp+70h+var_60]
```

又有明文出现,猜想又是比对。但之后汇编逻辑比较复杂,至此祭出F5神器。 int \_\_cdec1 sub\_401080(HWND hDlg)

int result; // eax@5
CHAR String; // [sp+4h] [bp-64h]@1
char v3; // [sp+5h] [bp-63h]@1
char v4; // [sp+6h] [bp-62h]@2

```
char u5; // [sp+8h] [bp-60h]@3
__int16 u6; // [sp+65h] [bp-3h]@1
char u7; // [sp+67h] [bp-1h]@1
  String = 0;
  nemset(&v3, 0, 0x60u);
  v6 = 0;
  v7 = 0;
 GetDigitenTextA(hDlg, 1000, &String, 100);
if ( v3 == 'a' && !strncmp(&v4, a5y, 2u) && !strcmp(&v5, aR3versing) && String == 'E' )
  ۲
    MessageBoxA(hDlg, Text, Caption, 0x40u);
    result = EndDialog(hDlg, 0);
  3
  else
  {
    result = MessageBoxA(hDlg, aIncorrectPassw, Caption, 0x10u);
  return result;
3
显示了整个该函数的逻辑,其中关键的一句:
if ( 03 == 'a' && !strncmp(&04, a5y, 2u) && !strcmp(&05, aR3versing) && String == 'E' )
{
   MessageBoxA(hDlg, Text, Caption, 0x40u);
   result = EndDialog(hDlg, 0);
)
```

显然当所有条件满足的时候,调用messagebox函数(即输出对话框,显示congratulations)。 分析可知即字符串由a,5y,R3versing,E构成,同时验证上述汇编级别的分析正确性。 对比这两部分。

```
int result; // eax@5
CHAR String; // [sp+4h] [bp-64h]@1
char v3; // [sp+5h] [bp-63h]@1
char v4; // [sp+6h] [bp-62h]@2
char v5; // [sp+8h] [bp-60h]@3
__int16 v6; // [sp+65h] [bp-3h]@1
char v7; // [sp+67h] [bp-1h]@1
```

String= byte ptr -64h var\_63= byte ptr -63h var\_62= byte ptr -62h var\_60= byte ptr -60h

根据偏移量可以得出 输入字符串由String+var\_63+var\_62+var\_60组合而成, 将这几部分组合起来,即得到了Ea5yR3versing,即为flag。 Ps: 不习惯汇编可以直接看F5反编译代码。

#### KeygenMe 100

运行知,本题为win32控制台程序,相对第一题没有众多apl函数,结构更清晰。 直接F5查看伪C代码,其中重要的逻辑为:

```
sub_4011B9((int)aInputName, v6);
scanf(aS, &u11);
U3 = 0:
for ( i = 0; v3 < (signed int)strlen(&v11); ++i )</pre>
Ł
  if ( i \ge 3 )
    i = 0;
  sprintf(&v15, aS02x, &v15, *(&v11 + v3++) ^ *(&v8 + i));
3
memset(&v11, 0, 0x64u);
sub_4011B9((int)aInputSerial, v7);
scanf(aS, &v11);
if ( !strcmp(&u11, &u15) )
Ł
  <mark>sub_4011B9</mark>((int)aCorrect, *(int *)&v8);
  result = 0;
3
else
{
   1
  <mark>sub_4011B9</mark>((int)aWrong, *(int *)&v8);
  result = 0;
3
return result;
```

其中sub\_4011B9显然是一个类似print的函数,那么第一部分: 先输入值到v9,再把v9经过异或运算后赋值给v13。

```
v6 = 16;
 v7 = 32;
 v8 = 48;
 sub_4011B9(aInputName);
 scanf(aS, &u9);
 v3 = 0;
 for ( i = 0; v3 < (signed int)strlen(&v9); ++i )</pre>
 ĸ
   if ( i \ge 3 )
    i = 0;
   sprintf(&u13, aS02x, &u13, *(&u9 + u3++) ^ *(&u6 + i));
 }
i>=3时候i=0就是i%3的运算,
但是v6+i表示的又是什么呢?
参看变量定义,可以知道v6,v7,v8为1个字节,v6+1即v7,v6+2即v8
char v6; // [sp+Ch] [bp-130h]@1
char v7; // [sp+Dh] [bp-12Fh]@1
char v8; // [sp+Eh] [bp-12Eh]@1
所以得到: V13[i]=v9[i]<sup>4</sup>v6[i%3];
第二部分:再次输入一个变量的值。
sub_4011B9(aInputSerial);
scanf(aS, &09);
下面就进行判断,如果v9和v13相等则输出congratulations,flag正确。
if ( !strcmp(&v9, &v13) )
{
  sub 4011B9(aCorrect);
result = 0;
log.csdn.net/yuriufo
 A1 CA
即V13[i]=v9[i]<sup>4</sup>v6[i%3]和5B,13,49,77,13,5E,7D,13进行对比
所以将Serial转成十进制后,编写脚本(不会python,只能用c++)
  int a = {91, 19, 73, 119, 19, 94, 125, 19 };
  int b[] = \{16, 32, 48\};
  for (int i = 0; i < 8; i++)
  {
      a[i] = a[i] <sup>^</sup> b[i % 3];
      cout << a[i] << ",";
  }
为flag的十进制数字,再转成char即得到flag。
```

 C:\Windows\system32\cmd.exe 75, 51, 121, 103, 51; 110; 109, 54g请按任意键继续u::iufo

#### **MAZE 200**

```
根据下载的文件,打开分析,应该是用C++写的一个程序,找到主函数的入口。
f1();
f2();
wprintf(L'Input Your (Key;\n'i);/yuriufo
scanf s("%s", &v11, 29);
在经过一些初始化后,发现主函数调用了2个重要的函数,于是跟进到f1。
srand(0xFFu);
hang = 0;
v3 = 15;
do
```

```
nany *= 1>;
--v3;
}
while ( v3 );
return rehult;://blog.csdn.net/yuriufo
}
```

显然该函数,使用rand()构造随机数,但是是伪随机数,每次生产的table都是一样的,从上述过程来 看,hang+=15即代表table中的点往下移动一格,而lie++则表示table中的点往右边移动一格。table的大小为 15\*15。

93	Dő	79	46	87	A4	CA	8E	ED	1F	ØA	64	C 0	14	86	7B	•••	,Ε.,	d{	
3E	<b>B8</b>	ØE	28	6B	22	76	E5	80	CE	18	CF	AC	10	51	27	>	. (k'	'vQ'	
1B	CF	00	D4	E9	8F	ØA	EØ	) 6F	67	D8	F7	C7	47	D7	EB			ogG	
4F	83	F7	33	2B	53	2D	67	C 6	51	F2	C5	39	FE	C1	AF	0.	.3+5	S-g.Q9	
05	3D	9D	2B	56	Dó	88	62	A4	F4	ØD	21	29	A7	B7	5B	-=-	.+0.	bt)[	
63	63	97	A5	95	7F	C3	B9	45	<b>B8</b>	D1	F4	BF	AC	60	D8	CC.		E`.	
91	5F	90	8A	ØE	<b>B8</b>	85	54	CB	05	E8	24	25	73	65	0C			T\$%se.	
<b>B8</b>	97	2E	CØ	E9	E9	76	18	5E	43	F7	9B	80	66	6D	EØ			.v.^C∎fm.	
00	75	19	31	4F	78	3F	- F7	24	DA	A9	3F	FB	EB	21	30	• u .	.10>	.\$?<b !<	
90	5F	FA	C3	68	CE	88	CE	48	32	A3	FA	BB	6C	29	09		. <u>.</u> h.	H21).	
91	BF	79	60	58	-53	F 8	88	- F 6	83	90	83	EB	50	20	2D	J	1 IS	SP,-	
28	FU	30	EF	51	10	38	11	45	00	10	52	81	FF.	03	92	*	•.ųc	08.EK	
84	/E	EF 0Z	58	21	88	- F U	40		20	00	BE E4	35	ET	05	TE DD	1	. AQ .	· · · · · · · · · · · · · · · · · · ·	
10	HU	30	00	E4	00	67	24	93 00	JE 00	8E	th:	10	PEC	HH	eg di	n. 1	iet	/)uriufo	
13	00								00				00	99	00				
OOAC	453	0	67	5A	DF	F8	EA	89 D	C B	3 2	2D 4	16 D	A C	4 08	8 21	EA	C7	gZF	
OOAC	;454	0	Eő	DD	F3	C9	46	F3 2	D CO	: 3	BB 1	0 A	5 4	8 E(	C 92	BC	A4	F;.	H
OOAC	:455	0	15	ØF	A8	C1	84	CE F	1 7	9 9	C 4	IC F	5 F	6 D	5 B8	65	8F	p.l	e.
OBAC	:456	0	3B	77	C6	E8	EB	A1 F	0 A1	4 9	PA 8	2 9	2 D	8 01	D 1A	AC	8F	;w	
OOAC	:457	0	A1	9D	14	47	C5	F4 F	2 72	2 7	<b>B</b> 3	A 4	4 F	6 DE	8 40	5B	AD	Gr{:	:D@[.
U U U U U	458	U	90	UA	58	E5	58	50 B	F E'		88 F	DD	3 5	0 80	5 83	38	FØ	[.XP	.08.
BBAC	459	0	4E	45	03	CB EE	CD	3B 1	E F		19 5	2 0	0 0	2 5	FH OD	OC OF	00	HEC;	·
BBAD	428	0	23	16	D6	8R	30	SF D	6 14		10 U	2 H	BE	2 90 C 91	3 25	PD PD	ES	WE = E	11 9
RRAD	450	ß	37	10	DB	76	88	R1 7	F 8	1	13 1	6 4	0 1	A 7	F 10	20	10	7	M~
ABAD	:450	ß	64	FØ	AC	6.9	F5	2F F	8 9	8 3	13 F	EF	4 4	4 8	5 81	9F	85	3	.D.
OBAG	45E	0	89	18	30	88	CB	E3 E	E 7	2	2F 2	7 6	18 E	0 70	C 66	FD	50		If.P
			_			-						-	-						
OBAC	45F	0	8E	54	25	C4	53	59 2	6 31	3 1	ιF 1	2 8	0 F	8 18	B CE	02	78	.T%.SY&00.	x
00A0 00A0	45F	0	8E DA	54 13	25 52	C4 7D	53 D5	59 2 17 6	6 3) A C	9 L 9 D	F 1	28	0 F 5 9	8 18 3 D2	8 CE 2 C4	02 76	78 05	.T%.SY&00.	×

第二部分:

接着跟进到f2函数,看起来运算比较复杂。又创建了一个desk,同样是15\*15。 其实所有的类似于byte\_404530的字节变量都是f1函数中的构造table中的每一个点。

<pre>hang = 0; u1 = 15; do { ilie = 0; do {</pre>	<pre>*(&amp;byte_40440[hang] + lie)) - *(&amp;byte_403280[hang] + lie); *(&amp;byte_404531[hang] + lie)) - *(&amp;byte_403281[hang] + lie); *(&amp;byte_404532[hang] + lie)) - *(&amp;byte_403282[hang] + lie); *(&amp;byte_404533[hang] + lie)) - *(&amp;byte_403284[hang] + lie); *(&amp;byte_404534[hang] + lie)) - *(&amp;byte_403284[hang] + lie);</pre>
hang +- 15; 1; > while ( v1 ); return lie; >	http://blog.csdn.net/yuriufo

一次循环创建5个点,lie+=5往右边移动5个点;hang+=15则是点往下移动一格。

得到的表如图所示:

1				-	-	-												
I	00AC4620	51	00	81	81	81	<mark>81</mark>	01	<mark>81</mark>	00	00	00	00	00	00	00	<b>81</b>	
	00AC4630	01	<mark>01</mark>	00	00	00	<mark>01</mark>	<mark>01</mark>	<mark>01</mark>	<mark>01</mark>	01	00	00	00	00	01	00	
	00AC4640	01	00	00	00	00	00	00	00	00	01	<mark>01</mark>	01	00	00	00	01	
	00AC4650	01	00	00	<mark>01</mark>	<mark>01</mark>	<mark>01</mark>	<mark>01</mark>	<mark>01</mark>	00	00	00	00	01	00	00	01	
	00AC4660	<mark>01</mark>	<mark>01</mark>	<mark>01</mark>	00	00	00	00	<mark>01</mark>	<mark>01</mark>	01	00	<mark>01</mark>	00	00	00	00	
	00AC4670	01	00	00	00	<mark>01</mark>	<mark>01</mark>	01	<mark>01</mark>	00	00	00	01	01	<mark>01</mark>	01	01	
	00AC4680	01	<mark>01</mark>	<mark>01</mark>	<mark>01</mark>	00	00	00	<mark>01</mark>	00	00	00	00	00	00	01	00	
	00AC4690	00	00	00	00	00	00	01	<mark>01</mark>	01	01	<mark>01</mark>	01	01	01	01	01	
	00AC46A0	01	<mark>01</mark>	00	<mark>01</mark>	01	<mark>01</mark>	00	00	00	00	00	00	<mark>01</mark>	00	00	00	
	00AC46B0	00	00	00	00	00	00	01	00	00	00	00	01	<mark>01</mark>	<mark>01</mark>	01	00	
	00AC46C0	00	<mark>01</mark>	01	<mark>01</mark>	00	00	00	00	00	00	00	00	00	<mark>01</mark>	01	01	
	00AC46D0	00	00	00	00	01	00	00	00	00	00	00	00	00	00	01	01	
	00AC46E0	00	00	00	00	00	00	00	00	<mark>01</mark>	01	<mark>01</mark>	01	01	00	01	00	
	00AC46F0	00	00	01	01	01	01	01	<b>01</b>	00	00	0.0	00	00	01	01	01	
	00AC4700	01	00	00	00	00	00	00	00	02	00	00	00	00	00	00	00	

之后主要在主函数中进行分析。 wprintf(L"Here is the flag!\n");

```
printf("00");
 v8 = 0;
 do
 {
   u9 = *(&u13 + u8);
   if ( 09 )
    printf("%02X", v9);
                                            T
   ++v8;
 ->
 while ( v8 < 30 );
 wprintf(L"\nPlease upload the MD5 of the correct flag!\n");
 system("pause");
 return 0;
可以看到,v11为输入的值,v11+v6为输入的每一个值。
While循环确定的是当每一个值为0x49,即字符1的时候,++lie,即向右移动一格。
当值是0x50的时候,lie+=15,即向下移动一格。
注意看LABEL 9部分:每次处理完输入的值之后,将值赋给v7,而lie是从0开始的,也就是说v7变量表示的是走
过的路径:
例如:
 01 00 00 00 00 00 00 00 32 31 31 32 32 31 32 31
                                                31 32 32 32 32 32 32 31
                       31 31 32 31 31 32 31 32
                                             1222222111211212
 32 31 31 31 00 00 00 00
                       00 00 00 00 00 00 00 00
                                              2111.....
 00 00 00 00 00 00 00 00
                        00 00 00 00 00 00 00 00
                                               . . . . . . . . . . . . . . . .
 00 00 00 00 00 00 00 00
                        00 00 00 00 00 00 00 00
                                               . . . . . . . . . . . . . . . .
 00 00 00 00 00 00 00 00
                       00 00 00 00 00 00 00 00
                                               . . . . . . . . . . . . . . . .
                       00 00 00 00 00 0F 10 11
 00 00 00 00 00 00 00 00
                                               . . . . . . . . . . . . . . . . .
 20 2F 30 3F 40 41 50 5F
                       6E 7D 8C 9B 9C/9D 9E AD
                                              / 0?@AP_n } . . . .
 AE AF BE BF CE DD DE DF E0 00 00 00 8D B0 5C 4B
                                               .....\K
```

2112212112222221112112122111序列表示我输进去的值,而下面从0F开始的值,容易发现,从0F-E0,一共 225,即一共走了225步,或者说从desk[0][f]-desk[e][0]。



但其实入口地址还没有呈现,即迷宫的入口在哪里。

其实无论你输进去什么值,程序默认都会打印出走的路径,注意到:

Input Your Key: 2112212112222221112112122111 Here is the flag! 000F1011202F303F4041505F6**B7D8C9E9C9D9BADAEAFBEBRCEDDDEDFEO** 

第一部分都会输出00。下面的while循环就是在打印之后的路径,所以,迷宫入口即为desk[0][[0]。 这时候就会发现,为什么一个二维迷宫只有2个方向键控制,因为是从左上角到右下角。 根据给的路径,只需要往下和往右就行了。

101111110000000
111000111110000
101000000001110
001100111110000
100111100001110
100001000111100
011111111100010
000001000000011
111111111101110
000001000000000
100001111001110
000000001110000
1000000000011000
000001111101000

# 1111110000011111

由于行列大小不一样,看起来有点奇怪,不是正方形。(0为墙,1为路)。 最后将得到的路径MD5(32位)之后,改成大写即为flag。

#### •

**PWN** 

#### ....

۴.

# 盲打(笑) **50**

根据题目在命令行中进行远程通信nc 202.119.201.199 28888。 然后根据提示进行输入,当长度不够的或长度过长的时候都有提示。 当长度接近的时候慢慢试。。。 最后长度确定后,提示更改最后4个字节,猜想是flag,于是得到flag。 (赛后端口好像挂了,所以没有截图。)

#### **Introductory Chapter 120**

这道题给了文件,下载下来,发现是栈溢出,浏览程序,发现只要将eip调用到此处即可:

text:0000000000400716	push	rbp
text:0000000000400717	nov	rbp, rsp
text:000000000040071A	sub	rsp, 30h
text:000000000040071E	nov	esi, offset unk_400884
text:0000000000400723	nov	edi, offset aCatFlag_txt ; "cat flag.txt"
text:0000000000400728	call	_popen
text:000000000040072D	nov	[rbp-8], rax
text:0000000000400731	lea	rax, [rbp-30h]
text:0000000000400735	nov	edx, 28h
text:000000000040073A	nov	esi, 0
text:000000000040073F	nov	rdi, rax
text:0000000000400742	call	_nenset
text:0000000000400747	nov	edx, 9
text:000000000040074C	nov	esi, offset aKazisa ; "kazisa!\n"
text:0000000000400751	nov	edi, 1
text:0000000000400756	call	_write
text:000000000040075B	nov	rdx, [rbp-8]
text:000000000040075F	lea	rax, [rbp-30h]
text:0000000000400763	nov	esi, 28h
text:0000000000400768	nov	rdi, rax
text:000000000040076B	call	_fgets
text:0000000000400770	lea	rax, [rbp-30h]
text:0000000000400774	nov	rdi, rax
text:0000000000400777	call	_strlen
text:000000000040077C	nov	rdx, rax/blog.csdn.net/yuriufo
text:000000000040077F	lea	rax, [rbp-30h]

注意到一下几个步骤,所以buf的位置为rbp-30h,

buf= byte ptr -30h push rbp

BOPog.csbp.rsp/yuriufo sub rsp, 30h

考虑到push rbp 产生的8个字节,实际要覆盖rip需要38h个字节,首先,为避免栈的不平衡,先输入48个字节的 值观察:

 61
 61
 61
 61
 61
 61
 62
 62
 62
 aaaaaaaaaaaabbbb

 62
 62
 62
 62
 62
 62
 62
 62
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63
 63

 00007FFD0CF56020
 616161616161616161

 00007FFD0CF56028
 6262626261616161

 00007FFD0CF56038
 62626262626262

 00007FFD0CF56038
 6363636363636363

 00007FFD0CF56048
 6464646463636363

 00007FFD0CF56058
 64646464646464

 00007FFD0CF56058
 09000000000000

 00007FFD0CF56058
 090007F80CB5ACF45

 11bc\_2.19.so:\_\_11bc\_start\_main+F5

其中00007FFD0CF56050为保存的rbp值,00007FFD0CF56058为rip返回值,上图说明想法基本正确。下面考

虑如何改变rip。 .text:000000000400716; .text:0000000000400716 push rbp .text:0000000000400717 mov rbp, rsp .text:0000000000400716 sub rsp, 30h .text:000000000040071E mov esi, offset unk\_400884 .text:0000000000400723 mov edi, offset aCatFlag\_txt<sup>O</sup>g.csdp...ot flag.txt<sup>O</sup>

主要到00000000400716为获取flag的函数,根据小段格式,输入的值应该为:

[\*] Opening connection to 202.119.201.199 on port 22666: Done
[\*] Switching to interactive mode
this is really easy!!!
Are you haluki?
kazisa!
\x00flag{the\_3ast\_get\_stach\_ov-\_-233}
[\*] Got EOF while reading in interactive
http://blog.csdn.net/yuriufo

•

剩下的题都不会做了,总和的writeup:CUMTCTF2017入门赛Writeup

转载于:https://www.cnblogs.com/yuriufo/p/7360847.html