攻防世界Crypto第一页-进阶区



<u>人生若只如初见Crypto</u> ● 于 2020-04-01 11:40:35 发布 ● 1343 ☆ 收藏 2 文章标签: <u>密码学</u> 版权声明:本文为博主原创文章,遵循<u>CC 4.0 BY-SA</u>版权协议,转载请附上原文出处链接和本声明。 本文链接: <u>https://blog.csdn.net/weixin_44159598/article/details/103346365</u> 版权

你猜猜



• 打开附件明显为 16 进制,查询得知为 zip 的文件开头

📲 578c186a88544d4e9b2552d474e04115.txt - 记事本	-		
2件(F) 編攝(E) 楷式(O) 查看(V) 帮助(H)			
04B03040A0001080000626D0A49F4B5091F1E00000	12000000800000666C61672E7478746C9F170E	035D04	44
26A03E161FB96870EDDFC7C89A11862F9199B4CD78	7504B01023F000A0001080000626D0A49F4B509	91F1E0	00
0120000008002400000000000000200000000000	0666C61672E7478740A00200000000000010018	00AF1	50
CAF2D1015CAEAA05CAF2D1015CAEAA05CAF2D101	04B05060000000010001005A00000044000000	00000	

• 打开 Winhex 后,新建文件,将内容导入保存为 1.zip,打开压缩包

1.zip											-						
Offset	0	1	2	3	4	5	6	7	8	9	A	в	С	D	E	F	ANSI ASCII 🔨
00000000	50	4B	03	04	0A	00	01	08	00	00	62	6D	0A	49	F4	B5	PK bm Iôu
00000010	09	1F	1E	00	00	00	12	00	00	00	08	00	00	00	66	6C	fl
00000020	61	67	2E	74	78	74	6C	9F	17	0D	35	DO	A4	58	26	AO	ag.txtlŸ 5ФX&
00000030	3E	16	1F	B9	68	70	ED	DF	C7	C8	9A	11	86	2F	91	99	> 'hpißÇÈš t/ 'm
00000040	B4	CD	78	E7	50	4B	01	02	ЗF	00	0A	00	01	08	00	00	'ÍxçPK ?
00000050	62	6D	0A	49	F4	B 5	09	1F	1E	00	00	00	12	00	00	00	bm Iôu
00000060	08	00	24	00	00	00	00	00	00	00	20	00	00	00	00	00	Ş
00000070	00	00	66	6C	61	67	2E	74	78	74	0A	00	20	00	00	00	flag.txt
00000080	00	00	01	00	18	00	AF	15	02	10	CA	F2	Dl	01	5C	AE	- ÊòÑ \⊗
00000090	22	05	CA	F2	DI	01	50	AF	22	05	CA	F2	DI	01	50	4R	· ÊAÑ \@· ÊAÑ PK

000000000	nn	00	U.A.		24	V.1	50	-	~~	00	U.A.		24	~+	90	10	201	10	FON	E 41
0A000000	05	06	00	00	00	00	01	00	01	00	5A	00	00	00	44	00			Z	D
00000B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
00000000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
00000D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0.0				
•											-									

• 此时发现压缩包有密码,使用 ziperello 爆破解密

	当前密码长度	<u></u>	6	· 建合物体、速石本 (步骤 4	
	当前密码	信息	×			
逝去时间: 00:00:00	0%	а	码: 123456	用1720号3X22大臣作 耗时较长。	17. 1 MA 40 41	지네포함가포 #아이 (
47.41. (202) . 14334641	etia: os					

• 解出flag

合 🔲 🧱 🕌 1.zip - 解包大小为 1 KB				
S称(上级目录)	压缩前	压缩后	<u> </u>	修改日期
a flag.txt *	1 KB	1 KB	文本文档	2016-08-10 1

flag.t	xt - 记事:	本			
文件(F) \$	编辑(E)	格式(O)	查看(V)	帮助(H)	

https://blog.csdn.net/weixin_44159598

enc

enc 最佳Writeup由admin提供	III WP ● 建议
难庭系数: ◆★1.0	
题目来源: 暂无	
题目描述:Fady不是很理解加密与编码的区别所以他使用编码而不是加密,给他的朋友传了一些秘密的消息。	
题目场景: 暂无	
题目附件: 附件1	

• 可以看到为 zero 和 one,明显为二进制,全部转换为 0 和 1,这时候可以利用 word 等专业文字编辑工具

 $\label{eq:light} Li0gLi0uLiAuIC0uLi0gLS4tLiAtIC4uLS4gLSAuLi4uIC4tLS0tIC4uLi4uIC0tLSAuLS0tLSAuLi4gLS0tIC4uLi4uIC4uLSAuLS0uIC4uLi0tIC4tLiAuLi4gLS0tIC4uLi4uIC4uLSAuLS0uIC4uLi0tLC4uLi4gLiAtLi0uIC4tLiAuLi4tLSAtIC0uLi0gLQ = = 0 \label{eq:light}$

https://blog.csdn.net/weixin_44159598

- 将二进制转为字符串
- 明显的 base64 编码, base64 解密,摩斯密码

LEXCTFTH1501S05	JP3R05ECR3TOTXT	
转换为摩斯电码	清除 生成摩斯代码的分隔方式: ● 空格分隔 ○ 单斜	: 料杠/分隔
謝电码: (格式要求: 	可用空格或单斜杠/来分隔摩斯电码,但只可用一种,不可混 -.-..-.-.-...--- --- .--- -....-- .-.-- -.-.-.-.)

• Flag 即为 ALEXCTF{TH15_1S_5UP3R_5ECR3T_TXT},最后一步的小坑,将 O 替换为"—"

告诉你个秘密

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

636A56355279427363446C4A49454A7154534230526D6843 56445A31614342354E326C4B4946467A5769426961453067

猜测为16进制,转换为字符串↔

36A563552794	127363446C4A49	54A7154534230526D684356445A31614342354E326C4B4946	6467A5769426961453067

Base64 解码↔

https://blog.csdn.net/weixin_44159598

	NOTURAL PARTICIPATION DEPART		
	r5yG lp9I BjM tFhBT6uh y7iJ QsZ bhM		
	BASE64加密 BASE64解密	h.	
发现	砚直接分组,看到键盘上每组字母之间有规律,所以为键盘密码↔		

解密后 flag 为 tongyuan↩ https://blog.csdr

Easy-one

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int main(int argc, char **argv) {
if (argc != 3) {
printf("USAGE: %s INPUT OUTPUT\n", argv[0]);
return 0;
FILE* input = fopen(argv[1], "rb");
FILE* output = fopen(argv[2], "wb");
if (!input || !output) {
printf("Error\n");
return 0;
char k[] = "CENSORED";
char c, p, t = 0;
int i = 0;
while ((p = fgetc(input)) != EOF) {
c = (p + (k[i % strlen(k)] ^ t) + i*i) & 0xff;
t = p;
i++;
fputc(c, output);
return 0
```

- 是使用 k[]和 input 经过加密算法后生成 output
- 文件 msg001.enc 和 msg001 是 对 应 的 , 应 该 是 用 k[] 和 msg001(input) 生 成 了msg001.enc(output),在这个加密过程 中,k[]好像是个常量,代码中给出的应该是个例子,不是真实的 k[],对 msg001 加密后并不是 msg001.enc,看来我们需要 先找到这个真实的 k[]了。
- 根据 msg001 和 msg001.enc, 把算法逆过来算出 k[]:

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int main(int argc, char **argv) {
if (argc != 2) {
printf("USAGE: %s INPUT OUTPUT\n", argv[0]);
return 0;
//FILE* input = fopen(argv[1], "rb");
FILE* input = fopen(argv[1], "rb");
if (!input) {
printf("Error\n");
return 0;
char c, p, t = 0;
int i = 0;
char w[] = "Hi! This is only test message\n";  //原来 input 中的值
unsigned int j = 0;
while ((p = fgetc(input)) != EOF) {
// printf("read %d", p);
for (j=31;j<125;j++) {
c = (p - (j ^ t) - i*i) \& 0xff;
if (c == w[i]) {
printf("%c",j);
i++;
break;
```

return 0;

编译之后对 main 传参:

D:\Reno\Desktop>demsg2.exe msg001.enc VeryLongKeyYouWillNeverGuessVeq>_

int main(int argc, char **argv)argc 表示参数个数,即是后面 argv 数组的元素个数,不用输入,会根据传入参数计算,只需传递 argv 数组的元素即可。argv 数组的第一个元素(argv[0])是程序名,传参的时候注意 argc 的值加上这个元素,其余的元素自行传递。这样就得到 k[]: VeryLongKeyYouWillNeverGuess,然后利用 k 再对 msg002.enc 进行解密,得到 msg002.enc 对应的明文:

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int main(int argc, char **argv) {
if (argc != 3) {
printf("USAGE: %s INPUT OUTPUT\n", argv[0]);
return 0;
FILE* input = fopen(argv[1], "rb");
FILE* output = fopen(argv[2], "wb");
if (!input || !output) {
printf("Error\n");
return 0;
char c, p, t = 0;
int i = 0;
char k[] = "VeryLongKeyYouWillNeverGuess";
i = 0;
c, p, t = 0;
int g = 0;
while ((p = fgetc(input)) != EOF) {
c = (p - (k[i % strlen(k)] ^ t) - i*i) & 0xff;
printf("Decrypting %x i=%d t=%d k=%d -> %d\n",p,i,t,(k[i % strlen(k)] ^ t),c);
t = c;
i++;
//printf("%c",c);
fputc(c, output);
g++;
if (g>450) {break;}
return 0;
```

555 管理员: C:\	\Windows\system32\cmd.exe	
D.\Reno\Des	kton)demsg3 eve msg002 enc msg002 tvt	
Decrypting	ffffffaa i=0 t=0 k=86 -> 84	
Decrypting	$ffffff0a i=1 \pm 84 = 49 \rightarrow 104$	
Decrypting	$ffffff83 = 2 \pm 104 = 26 - 101$	
Decrypting 4	$45 = 3 \pm 101 \text{ k} = 28 \rightarrow 32$	
Decrypting	ffffffe7 i=4 t=32 k=108 -> 107	
Decrypting f	ffffff8h i=5 t=107 k=4 \rightarrow 110	
Decrypting f	ffffff93 i=6 t=110 k=0 \rightarrow 111	
Decrypting f	ffffffh0 i=7 t=111 k=8 \rightarrow 119	
Decrypting f	ffffffea i=8 t=119 k=60 -> 110	
Decrypting f	ffffff89 i=9 t=110 k=11 -> 45	
Decrypting ?	28 i=10 t=45 k=84 -> 112	
Decrypting e	e i=11 t=112 k=41 → 108	
Decrypting f	ffffffff i=12 t=108 k=3 -> 97	
Decrypting ?	26 i=13 t=97 k=20 -> 105	
Decrypting 7	70 i=14 t=105 k=62 -> 110	
Decrypting 5	5c i=15 t=110 k=7 -> 116	
Decrypting 7	7d i=16 t=116 k=24 -> 101	
Decrypting f	ffffffa2 i=17 t=101 k=9 -> 120	
Decrypting f	ffffffee i=18 t=120 k=54 -> 116	
Decrypting f	ffffff9a i=19 t=116 k=17 -> 32	Record And Antalian (Astronomy)
Decrypting 4	47 i=20 t=32 k=86 -> 97	s strien(o)] " ();c);
Decrypting 3	31 i=21 t=97 k=4 -> 116	
Decrypting 5	5e i=22 t=116 k=6 -> 116	
Decrypting f	ffffffa5 i=23 t=116 k=51 -> 97	
Decrypting f	ffffffb7 i=24 t=97 k=20 -> 99	
Decrypting f	ffffffe2 i=25 t=99 k=6 -> 107	
Decrypting f	ffffffdc i=26 t=107 k=24 -> 32	https://blog.csdp.pat/weivin_4/159598
Decrypting a	54 i=27 t=32 k=83 \rightarrow 40	https://biog.csun.net/weixin_44155556

 最终得到明文: The known-plaintext attack (KPA) is an attack model for cryptanalysis where the attacker has samples of both the plaintext (called a crib), and its encrypted version (ciphertext). These canbe used to reveal further secret information such as secret keys and code books. The term"crib" originated at Bletchley Park, the British World War II decryption operation. The flag is CTF{6d5eba48508efb13dc87220879306619}駕牗[?顆剝.}9 壶?F*損犄溦烇?5 蹈

说我作弊需要证据

• 使用 wireshark 打开所给流量包后,进行 tcp 流追踪

▲ Wireshark · 追踪 TCP 流 (tcp.stream eq 0) · Basic-06.pcapng	-		×
Wireshark - iBBTCP IE (tcp.stream eq 0) - Basic-06.pcapng U0VR ID0gMTM7IERBVEEgPSAweDNiMDRiMjZhMGFkYvRhNmY2N2MyNmJiMGMIZDZMOyBTSUcgPSAweDJINUFiMjRmOWRjMjFkZjQwNmE4N2RI U0VR ID0gMDsgREFUQSA9IDB4MrkOtGIMDGvXTMsMADAxhjAy2GNiNTY5CGY0HjhiNEv7IFNJRyA9IDB4MvLvYjZkNGVhZjExNTgyZTgIZWFkNGJm U0VR ID0gMDsgREFUQSA9IDB4MmVyOTEIMGYxZTNxHWVMDIIY2IMDY3RVhY0W7IFNJRyA9IDB4MVLvYjZkNGVhZjExNTgyZTgIZWFkNGJm U0VR ID0gMJsgREFUQSA9IDB4MmVyOTEIMGYxZTNxHWVMDIIY2IMDY3RVhY0W7IFNJRyA9IDB4MTV2HWZIMMRhHzYYYZRKZTg5ZjI3WHM U0VR ID0gMjsgREFUQSA9IDB4MmVyOTEIMGYXZTNxHWVMDIIY2IMUY3DQWTFNJRyA9IDB4MTV2HWZIMMRhHzYYYZRKZTg5ZjI3WHM U0VR ID0gMjsgREFUQSA9IDB4MmVyOTEIMGYXZTNxHWVMDIIY2IMUXH2gIXHWVJFNJRyA9IDB4MTY3HWZHWZIMRRhHzYYYZRKZTg5ZjI3WHM U0VR ID0gMjsgREFUQSA9IDB4MmVyOTEIMGYXZTNXHWVMDIIY2IMUXH2g3YkW7IFNJRyA9IDB4MTM30GMyNTA00GMxOTgIH2I200E32WI5 U0VR ID0gMjsZTERBVEEgPSAweDESNJg4ZjEXMEE2NTEIXINHTXHTYSTWXTRWTTNY U0VR ID0gMjgZIERBVEEgPSAweDESNJg4ZjEXMEZXTEX0MSMDKwYTRmOTKxOGRMOyBISUcgPSAweDE0HDhhYZ2IZWUYYJIOTFNGE2MU20XZ U0VR ID0gMjgZIERBVEEgPSAweDSSD20AyHjRkNGExhZmYTVhOTEIMjFiNJYINDXDWJBSUcgPSAweDE0HDhhYZIZWUYYJIOTFNGE2MU20XZ U0VR ID0gMjgZIERBVEEgPSAweDUS2DAYHJRKNGExhZmYTVhOTEIMjFiNJYINDXDWJBSUcgPSAweDIIV2NIJjKANTE4YmZIZMEYNDY U0VR ID0gMjZ7IERBVEEgPSAweDUS2DAYHJRKNGExhZmYTVhOTEIMjFiNJYINDXDWBISUcgPSAweDIIVINIJJYJYJMAMTE4YmZIZMEYNDY U0VR ID0gMjZ7IERBVEEgPSAweDUYZDAYHTRKOVYYJXYZU1V2/FINJSYA9IDB4MjA4YmFIZOQ2HJMATTE4YmZIZMEYNDY U0VR ID0gMtZ7IERBVEEgPSAweDUAGUZNMNINTKSNzg20DFmODk2ZTM4ZDV1Y2FMOyBISUcgPSAweDJIMUIMjNJJWMTHYD0HDJK0WVINjBm U0VR ID0gMtQ7IERBVEEgPSAweDUYZTgYYZYFYJYJXYTKFNTJYZYMYZYMYSZMJWBJBSUcgPSAweDJMUUNZU0MTIYNDQ0HDJK0WVINjBm U0VR ID0gMtZ7IERBVEEgPSAweDIYZFMYXDYZYYXYXYXYXYXYXYXYXYXYXYXXFKNZJ1N0XHXQZSYLJ0YYZZJMWTKXYZZZGSUSXWJU U0VR ID0gMtZ7IERBVEEgPSAweDZYZZYHXYZZYFYYJZZYMYFKNTZJYWYZDYYYZJWBJDB4NZCHMMQ0NDNINZAYNTASYTZINDCXHjQZ U0VR ID0gMtZ7IERBVEEgPSAweDZYZZYJNXFKNTZYZYMYZXYXXFKNTZJ1N0XPBJDB4NZCHMMQKXYZXZZCGSUXDHJQVZZ U0VR ID0gMtZ7IERBVEEgPSAweDJBAHNYYYYZYFYYZYEFYYZXXXFKNTXZJYMYZMYYZYYMZYDNBABNJDB4NZCHMMKXYZZXZZWYZZZZGSUXTUQYZZ U0VR		MOW== MOW== T T T T T T T T T T T T T T T T MOW== T T T MOW== MOW== MOW== MOW== m MOW== m MOW== m MOW== m MOW== m m MOW== m m T T	×
UGVRILOBJENDSBREFUGSA9IDBAHIBRAZE2001MTTJJCHNEMBQIYZEWZOFKOWCKYWZIFNJKYA9IDBAHIIINIAKZE52/GYDWRMENDZAYG 2 UGVRIDOBJES 28REFUGSA9IDBAHIBRAZE2001MTJJMZKHZEGYZJYZAZIZYMZIYNIKWYZIKNJKWZIFNJKYA9IDBAHZINIXYIYSZGHYDWRMENDZYTG UGVRIDOBJMZMZIERBVEEBPSAWEDJJMDKIYMZKHZEYZYDHMMZNGZKOGEYMZMKMRMOYBTSUCBPSAWEDNKOTKSYZKONZKOTNIZMMARZRKOTG UGVRIDOBJMZMZIERBVEEBPSAWEDJJMDKIYMZKHZFYZYDHMMZNGZKOGEYMZMKMRMOYBTSUCBPSAWEDNKOTKSYZKONZKOTNIZMMARZRKOTG UGVRIDOBJMYZTIERBVEEBPSAWEDJJMDKINIKINJKJONNINDINZJMZXNOTSJKOGEYMZMKMROYBTSUCBPSAWEDNKOTKSYZKONZKOTNIZMMARZRKOTG UGVRIDOBJMJYZIERBVEEBPSAWEDJJYZCXNKE4ZTNJMKFKYJJZMCJMZIJSBUOHIDOBMH90OTSUCBPSAWEDDWMZGJJODKYTUZMJZIZ UGVRIDOBJMJYZIERBVEEBPSAWEDJYJCZNNE4ZTNJMKFKYJJZMCSNZ UGVRIDOBJMJYZIERBVEEBPSAWEDJYZCXNKE4ZTNJMKFKYJZMZNYZZOGZNOJBISUCBPSAWEDDWMZCMJZMZWQYZJCZOWZMOWUJNZUXYZNJ UGVRIDOBJMJFZIERBVEEBPSAWEDZNJFINJELZTNJMKFKYJZMZYMZZZOGINODFINTEZGGOLWOBTSUCBPSAWEDDWMZCMJZMZWWZTJNJRKNYONTM UGVRIDOBJKJFZIERBVEEBPSAWEDKINTZONCHKIJCHZUJYYZMZYMZZCGGINODFINTEZGGOLWOBTSUCBPSAWEDZZNICHWIMZUXTISNJRKNYONTM	Y JY 2HOW Y JNINEW NzhjZEW NWYYZ J1 Y JIXTDS NWNKMEW Mjc0Mzd NT11MkW OTM2M2E	17 17 17 17 17 17 17 11 10 10 10 10 10 10 10 10 10 10 10 10	

UØVRIDØgMTY7IERBVEEgPSAweD UØVRIDØgMjU7IERBVEEgPSAweD UØVRIDØgNjsgREFUQSA9IDB4NW UØVRIDØgNDsgREFUQSA9IDB4ND UØVRIDØgNDsgREFUQSA9IDB4ND	NiNGVjM2M5YTg0NmM00 I2ZGVi2Dk1MTBjMT2m QwNmI2ZDg0YTIwZmI2 QxYTYyYWI0Nz1kMjkz NiZGE0YmY5Yi0SOGE2	ZWQ4NTFŔMD1hY2U YmVkNGY2MjY0ZTh NzI0NDI0M2Y2NjJ YTNjM2QXMWQ2NWZ OG00Y2I2NWJMZIZ	xMjJMOy iNjBMOy mM0w7IF kZUw7IF mYZBMOv	ITSUCgPSAweDMwM2FHYjY3Zj ITSUCgPSAweDM1MDRkOWRKMT IJRyA9IDB4MTQ0OGFjNmVlZT IJRyA9IDB4YZA0MGZiMmQ1ZT ITSUCgPSAweGR1YmR1ZTZ1M2	A3ZjljYTE5NZYYNZk0MTBmYT k2OTVlZDgxZjdhOGZjOGNiOG JiMmUSMWEWYTYYNDFlNTkwZU k2OGMAMWRjOGIXNWJKNJliTD ZlZWYSMmVIMiU2MmEiNmVMN0	3MOw== ZMOw== w7 s= w7
整个对话(17 kB)	~	显示和保存数据为	ASCII	Y		液回
查找:						查找下一个(1)
			-	ALL NO. DUDING DUDING	SSM NOTWOX 1115-14	Deltable

• 为 base64 加密后的数据, 解密

SEO 0x2e5ab24f9dc21df406a87de0b3b4L; 0x7492f4ec9001202dcb569df468b4L; = 5; DATA = SEQ 0x1e3b6d4eaf11582e85ead4bf90a9L; 0x1b8bdf9468f81ce33a0da2a8bfbeL; 0x198901d5373ea225cc5c0db66987L; SEQ 0x1448ac6eee2b2e91a0a6241e590eL; 0x208babd43638118bfbfa24675ee9L; 0x2b15275412244442d9ee60fc91aeL; SEO 0x3d939c9477d93bfc83dd97c5f2f9L; 0x400a19b82a4700ffc8a7515d7599L; CEO - 0. DATA -

```
= 13; DATA = 0x3b04b26a0adada2f67326bb0c5d6L; SIG
                                                             SEO = 0: DATA =
                           SIG = 0xc9107666b1cc040a4fc2e89e3e7L
                            0x94d97e04f52c2d6f42f9aacbf0b5L; SIG
                                                             -
                                                SEQ = 4; DATA =
0x2c29150f1e311ef09bc9f06735acL; SIG = 0x1665fb2da761c4de89f27ac80cbL
SEO = 18; DATA = 0x181901c059de3b0f2d4840ab3aebL; SIG
                                                             =
                                                SEQ = 2; DATA =
0x8a03676745df01e16745145dd212L; SIG = 0x1378c25048c19853b6817eb9363aL
SEQ = 20; DATA = 0x674880905956979ce49af33433L; SIG =
                                                SEQ = 0; DATA =
0x633282273f9cf7e5a44fcbe1787bL; SIG = 0x2b15275412244442d9ee60fc91aeL
    = 28; DATA = 0x19688f112a61169c9090a4f9918dL;
                                                       SIG
                                                             -
                                                SEQ = 24; DATA =
0x59d0264d4a134fa5a91521b25e46L; SIG = 0x2bc3bf947c0e85444aa13efa1c15L
SEQ = 21; DATA = 0xd24562795754da7abe213ffc11eL; SIG
                                                             SEQ = 19; DATA =
0x75c1fbc28bb27b5d2db9601fb967L; SIG = 0x2b5b628bf8183400cdab7f5870b1L
SEQ = 33; DATA = 0x580e36ce59978681f893e38d5ecaL; SIG
                                                             -
                                               SEQ = 27; DATA =
0x1eea254d861b2dc7ec03b37ef9fbL; SIG = 0xd6268f00fe0e2964d56458f59e2L
    = 24; DATA = 0xa02a43cdf9aa345fe83f059cab4L; SIG
                                                             =
                                                SEQ = 4; DATA =
0x2edb62eac7c6e83082387da0576eL; SIG = 0x77d2d083e702509a6b471242fedL
SEQ = 14; DATA = 0x83afae83c1db7776751d56c3f09fL; SIG
                                                             SEQ = 5; DATA =
0x7ccc3d3cb267d75acf0b10f579ecL; SIG = # 0.0x26256f0cdc63fb0913051c9b9b4fL
                           0.20dabd0E10a10fbad4fc204a0bc01 CIC
```

• 发现前面为编号,后面为传输的数据即 DATA,题目所给 RSA 参数,求出各自的 d,p,q即可解出 flag

```
from Crypto.PublicKey import RSA
import gmpy2
import base64
A_n = 1696206139052948924304948333474767
A_p = 38456719616722997
A_q = 44106885765559411
#Bob's
B_n = 3104649130901425335933838103517383
B_p = 49662237675630289
B_q = 62515288803124247
A_{phi} = (A_p - 1) * (A_q - 1)
B_phi = (B_p - 1) * (B_q - 1)
A_d = int(gmpy2.invert(e, A_phi))
B_d = int(gmpy2.invert(e, B_phi))
A_rsa = RSA.construct( (A_n, e, A_d) )
B_rsa = RSA.construct( (B_n, e, B_d) )
```

```
flag = {}
with open('zero_one') as f:
   for s in f.readlines():
       line = str(base64.b64decode(s), encoding = 'utf8')
       seq = int(line.split(';')[0].split(' ')[2])
       data = int(line.split('0x')[1].split('L;')[0], 16)
       sig = int(line.split('0x')[2].rstrip('L;\n'), 16)
       decry = B_rsa.decrypt(data)
       signcheck = A_rsa.sign(decry, '')[0]
       if signcheck == sig:
           flag[seq] = chr(decry)
dic = sorted(flag.items(), key = lambda item:item[0]) #对字典按键值进行排序,返回值为列表
print(dic)
for i in dic:
                                                    https://blog.csdn.net/weixin_44159598
print(f)
```

Flag 为 flag{n0th1ng_t0_533_h3r3_m0v3_0n}

X_xor_md5

010	55	30	49	4C	56	D2	73	70	12	45	AS	BA	85	C0	3E	53	UOI	L/	/Òsp 1	e-•_j	À>S
020	73	1B	78	2A	4B	E9	77	26	5E	73	BF	AA	85	9C	15	6F	s x	*1	(éw&^:	SZ	2 0
030	54	2C	73	1B	58	8A	66	48	5B	19	84	B0	80	CA	33	73	T,s	2	ŠfH["°€!	Ê3s
040	5C	52	0C	4C	10	9E	32	37	12	0C	FB	BA	CB	8F	6A	53	\R	L	ž27	û°Ë	jS
050	01	78	0C	4C	10	9E	32	37	12	0C	FB	BA	CB	8F	6A	53	x	L	ž27	û°Ë	jS
060	01	78	0C	4C	10	9E	32	37	12	0C	FB	BA	CB	8F	6A	53	x	L	ž27	û°Ë	jS
070	01	78	0C	4C	10	9E	32	37	12	00	89	D5	A2	FC			x	L	ž27	\$Č¢i	ä

由于题目名称中提示了 XOR, 并且序列 01 78 0c 4c 10 9e 32 37 12 0c fb↔ ba cb 8f 6a 53重复出现, 所以考虑 XOR KEY就是↔

bash 01 78 0

滚动鼠标轴或单击, 开始截长图 4c 10 9e 32 37 12 0c fb ba cb 8f 6a 534

注意到 hMMM的大小写是反着的,并且 rctf后面应该是"{",所以 XOR KEYe 的每一个字节还应该再 xor 0x20,得到e

bash 00000000: 48 6d 6d 6d 2c 20 67 6f 6f 64 20 6a 6f 62 2c↔ Oa Hmmm, goodjob, 00000010: 74 68 65 20 66 6c 61 67 20 69 73 20 6e 6f↔ 74 20 theflagisnot 00000020: 52 43 54 46 7b 57 65 31 6c 5f 64 30 6e↔ 33 5f 1c RCTF{Well_dOn3_. 00000030: 75 74 5f 77 68 34 74 5f 69 35 5f↔ 2a 6b 65 79 00 ut wh4t i5 *key. 00000040: 7d 0a 20 20 20 20 20 20 20 20 此时的 XOR KEY=21582c6c30be1217322cdb9aebaf4a59,根据题目名称来看这是↔ 个 MD5, 在 https://hashkiller.co.uk/md5-decrypter.aspx查询到对应原文↔ 是"that"↔

替换 FLAG中的 key得到最终 FLAG↩

 $RCTF \{Well_d0n3_6ut_wh4t_i5_that\} \leftrightarrow$

https://blog.csdn.net/weixin_44159598

Easy_Crypto

 附件.txt - 记事本 文件(F) 编辑(E) 相式(O) 查看(V) 帮助(H)
 get buf unsign s[256]

get buf t[256]

we have key:hello world

for i:0 to 256

set s[i]:i

for i:0 to 256 set t[i]:key[(i)mod(key.lenth)]

```
for i:0 to 256
set j:(j+s[i]+t[i])mod(256)
swap:s[i],s[j]
```

```
for m:0 to 37
set i:(i + 1)mod(256)
set j:(j + S[i])mod(256)
swap:s[i],s[j]
set x:(s[i] + (s[j]mod(256))mod(256))
set flag[m]:flag[m]^s[x]
```

fprint flagx to file

https://blog.csdn.net/weixin 44159598

• 中间有的地方 i、j 没有给出值,统统初始化为 0 即可,解密脚本:

```
-*- coding: utf-8 -*-
 = open('enc. txt', 'r', encoding= ' ISO- 8859-1' )
C = f.read()
t=[]
key = 'hello world'
ch =
j=0#初始化
s = list(range(256)) #创建有序列表
for i in range(256):
j =(j+ s[i] + ord(key[i % len(key)])) % 256
s[i],s[j] = s[j],s[i]
i=0#初始化
j=0#初始化
for r in C:
i=(i+1)%256
j=(j+s[i])% 256
s[i], s[j] = s[j], s[i]
x=(s[i]+(s[j]%256))%256
ch += chr(ord(r) ^ s[x])
print(ch)
```

EIS{55a0a84f86a6ad40006f014619577ad3}

cr2-many-time-secrets

 题目没有任何提示,查找writeup后发现利用One Time Pad的重用导致的攻击。我首先把密文直接放到了CyberChief里看看 能不能解密。用了Magic模式并不能直接得到明文。因为OTP是利用明文XOR密钥得到密文的,我又尝试了XOR
 bruteforce,也不
 bruteforce,也不 **形心还地行**判为义。

• 对于OTP密码的重用,我们可以利用Crib dragging attack来破解。这是一种已知部分明文的攻击,counter mode的block cipher如果重用了IV或者counter也可以用这种攻击。具体的解释如下:http://travisdazell.blogspot.com/2012/11/many-time-pad-attack-crib-

drag.html,实现这种攻击的脚本: https://github.com/SpiderLabs/cribdrag,利用这个脚本来破解题目中的密文,首先把原文件中 有换行的密文合并成一行,把这行密文放入脚本中:

52a182239373d6f740a1e3c651f207f2c212a2	247f3d2e65262430791c263e203d63232f0f20653f2
07f332065262c31683137223679182f2f37213	3202f142665212637222220733e383f2426386b
Your message is currently:	
A	
49	
80	
120	
1/0	
100	
200	
240	
280	
Your key is currently:	
0	
40	
80	
120	
160	
200	
249	
290	
Please enter your crib:	https://blog.csdn.net/weixin_44159598

• 程序会提醒我们输入一个可能存在于明文或者密钥里的字符串,根据题目提示,flag的开头是 ALEXCTF{,把这串字符输入:



• 可以看到 0 这个选项就是有意义的字符串。对于可能有意义的字符串,程序会在序号之前加上***。程序提示输入正确的位置,我们输入 0。程序又会提示我们输入我们的 crib 是明文中的还是密钥中的,假设 flag 是密钥,就输入 key:

Enter t Is this Your me	e correct position, 'none' for no match, or 'end' to quit: 0 crib part of the message or key? Please enter 'message' or 'key': key sage is currently:	,
0	Dear Fri	
40		
80		
120		
160		
200		
240		
280		
Your ke	is currently:	
0	ALEXCTF{	
40		
80		

120		
160		
200		
240		
280		
Please	e enter your crib:	

• 这样程序就恢复了一部分明文。在刚才的结果中,不止0一个位置是有意义的,***260:"ncryptio"也同样有意义。再次输入 ALEXCTF{,输入260作为正确的位置。现在的结果如下:

Enter the correct position, 'none' for Is this crib part of the message or ke Your message is currently: 0 Dear Fri	no match, or 'end' to quit: 260 y? Please enter 'message' or 'key': key
200 240ncryptio 280 Your key is currently: 0 ALEXCTF{	
80 120 160 200 240ALEXCTF{	
280 Please enter your crib:	

- 回到明文开头,我们可以猜测这是一封信的开头,Fri开头的单词很可能是Friend。输入"Dear Friend,"作为 crib。得到 0: "ALEXCTF{HERE"。
- 根据 flag 的常见格式,可以猜测 HERE 之后是下划线。将"ALEXCTF{HERE_"作为 crib 输入,得到有意义的字符串有: ***
 260: "ncryption sch", *** 234: "gree with me ", *** 208: "cure,Let Me ", *** 182: "ever if the k", *** 156: " proven to be", ***
 130: "hod that is m", *** 104: "is the only e", *** 78: "n scheme, Ih", *** 52: "sed One time ", *** 26: "nderstood my "
- 先看 260,可以猜测后面的单词是 scheme,输入"ncryption scheme "作为 crib: 260: "ALEXCTF{HERE_GOES",将新的 key 后面加上下划线输入: *** 260: "ncryption scheme a", *** 234: "gree with me to us", *** 208: "cure, Let Me know", *** 182: "ever if the key is", *** 156: " proven to be not ", *** 130: "hod that is mathem", *** 104: "is the only encryp", *** 78: "n scheme, I heard ", *** 52: "sed One time pad e", *26: "nderstood my mista",* 0: "Dear Friend, This "52 的后面几乎可以确 定是 encryption, 而且这样填充的字母多,所以这次输入"sed One time pad encryption":52: "ALEXCTF{HERE_GOES_THE_KEY}

OldDriver

打开附件,发现为 RSA 的模型,应用费马定理解密↔

{"c":

73660675747411714617220651332429160804955059136632503300827474653836768939704114765507483948 41437418105312353971095003424322679616940371123028982189502042, "e": 10, "n": 25162507052339714421839688873734596177751124036723831003300959761137811490715205742941738406 548150240861779301784133652165908227917415483137585388986274803},

["c":

21962825323300469151795920289886886562790942771546858500842179806566435767103803978885148772 139305484319688249368999503784441507383476095946258011317951461, "e": 10, "n":

23976859589904419798320812097681858652325473791891232710431997202897819580634937070900625213 218095330766877190212418023297341732808839488308551126409983193}, ["c":

55696894202740669578359833905835852865700876190481101411877005841937926952354050778115443551 59290382357149374107076406086154103351897890793598997687053983, "e": 10, "n":

 $18503782836858540043974558035601654610948915505645219820150251062305120148745545906567548650\\ 191832090823482852604346478335353784501076761922605361848703623\},$

["c":

45082461680445135184524938827135363906367415415518058217903389737976159712718672485843798131 14125478195284692695928668946553625483179633266057122967547052, "e": 10, "n":

23383087478545512218713157932934746110721706819077423418060220083657713428503582801909807142 802647367994289775015595100541168367083097506193809451365010723}, ["c":

22966105670291282335588843018244161552764486373117942865966904076191122337435542553276743938 B17686729554714315494818922753880198945897222422137268427611672, "e": 10, "n":

31775649089861428671057909076144152870796722528112580479442073365053916012507273433028451755

436987054722496057749731758475958301164082755003195632005308493),/blog.csdn.net/weixin_44159598

• 脚本如下:

import gmpy2
def broadcast(n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10): n = [n1,n2,n3,n4,n5,n6,n7,n8,n9,n10] c = [c1, c2, c3, c4, c5, c7, c7, c10]
N = 1
for i in n:
N *= 1
N1 - []
for i in m:
N1.append(N/1)
7-0
for i in xrange(10):
T.append(long(gmpy2.invert(Ni[i],n[i])))
X =0
for 1 in xrange(10):
$X \leftarrow c[1] * Ni[1] * T[1]$
n3 = X\N
m = gmpy2.iroot(m3,10)
print m
c1=73668075747411714617220651332429160084955059136632583300627474653830768939784114705597489348414374181853123597109509342432267961694857112302898218950242 n=251625076223371442183966887373459617775112483672383180330995761137811480715205742941738466548150246861779301784133652165908227917415483317555338986274803 c2=216628255231804641307993208129976818586527999477154685839804217890656643570710380397888518477213938464136882493189955017844515778337695549405391317991144 n=25976859589964197993208129976818586522594773918912327104139720289781851687761398037668771992124180222974173288834946959551126409863193 c2=5656969443079903208129976818586552059709719109123710431997202878155063403707080525132169953376687719921241802229734173288834946959551126409851193 c2=56569694202740605573590339058335286570097619044110411877005419379205253540557781154435511463702690323371443741877040608154103551078997955079397831
htp=18093702016636540043974550855001654110969055905465239201302531062395120140745549090625754865019182209082246350498454783553270450167905122665301641870052 Ce4=5602+6040+451515146524908271355805001674541551550521709310577777075557512108724654779061214124778035554704655162548177095120697719032064577212987547552 Pd=233830874785455122187131579329347401107217060190774234180402200816577134205813582801509807142882647367994289759155551065411083078030975061938064511655010723
C=22200109/0212033330000491024450352/04400735001910742097049/04976191233743592450322774333001760922934143154940189227530019094097222212327432040492550017654300012556001555430001255600155543000125560015554300010000000000000000000000000000
c6-17963313063405945742968136916219038352135561785389534381262979264585397896844470879023686580540355160998533122970239261072020689217153126649390825646712087
n6 • 2224634202299343322200001904441556052099283706538411726322932278881744954022406330010100155726421205045911087700330919213945046074833823905521643023679033949
C/1025/41/33/109279104053/00532/10232070011/01297/3046/392200314000000/402300402490310133/909622/(774050303140231004310473020249500102473004030403403403403403403403403403403403
c8=15585771734488351039456631394040497759568679429510619219766191780807675361741059290490732451112648776648126779759368428205194684721516497026290961786219352
n8+3205650859274418490128941328772803989130382331154860814108822787632675367415412477513277692848193537818475675678510754678163257029533864867382288173167869947
CM #99031234210379949090444310844023379105341476027312481298320324423900048174027462900481407467080210810977342254941014094570007467990602339797 mm 52584578054110747237811040421312481206446739105402541448090983437425303044431900681740114028737378111011757371101014519737814461990731453
c10=13560945756543023008529388108446048847137853838437995244573035888531288577370829005656532000539789839488848484840470303210189156383818339535589583427719989978247
n10-30415984800307578932946399987559088968355638354344823359397204419191241802721772499486615661699680996302439901585573958089047918537906687840725805496238621
broadcast(n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10)
prepulsion and polytopic 1/450500

• 改为 16 进制后转字符串即为 flag

6c61677b776	130517468335174	47234696e5f69355f6c6561	76316e675f6733745f6f6e5f69747d
地戰至符	字符转16进制	清空结果	
进制转字符 wo0 th3 tr	字符转16进制 4in i5 leav1ng g	清空结果 at on it)	
进制转字符 (wo0_th3_tr	字符转16进制 4in i5 leav1ng_g	清空结果 (3t_on_it)	

wtc_rsa_bbq

• 附件 winhex 打开后为压缩包开头,修改后缀,有两个文件: cirpher.bin 和 key,pem,直接使用 RsaCtftool 工具得到 flag

root@kali: ~/RsaCtfTool									0	•	6	
文件(F)	编辑(E)	查看(V)	搜索(S)	终端(T)	帮助(H)							
root@ka	li:∼/Rs bin	aCtfToo	l# pyth	ion Rsa(tfTool.	pypu	blickey	key.pem	uncip	her	fil	e
/usr/li ing: ur Reque	b/pytho llib3 (stsDepe	n2.7/di 1.25.3) ndencyW	st-pack or cha (arning)	ages/re rdet (3	equests/ 8.0.4) (/init_ loesn't	py:91 match a	: Reques support	tsDepend ed versi	end on!	∶y₩a	rn
[+] Cle this_On	ar text ?}	: Cong	ratulat	ions! H	Here is	a treat	for yo	u: flag{ :文件天	how_d0_y Rsactito	ou	7ur	

cr4-poor-rsa

• 首先我们看到文件中有 key.pub 文件,我们可以使用 openssl 来解析它,指令为:



from Crypto.PublicKey import RSA import gmpy2, base64 pub = open("key.pub", "r").read() pub = RSA.importKey(pub) n = long(pub.n) print "n" print n e = long(pub.e) print "e" print e #w/ n, get p and q from factordb.com p = 863653476616376575308866344984576466644942572246900013156919 print "p" print p q = 965445304326998194798282228842484732438457170595999523426901 print "q" print q d = long(gmpy2.invert(e,(p-1)*(q-1))) print "d" print d key = RSA.construct((n,e,d)) secret base64.b64decode("Ni45iH4UnXSttNuf0Oy80+G5J7tm8sBJuDNN7qfTIdEKJow4siF2cpSbP/qI WDjSi+w=") print key.decrypt(secret)

• 结果为: ALEXCTF{SMALL_PRIMES_ARE_BAD}

flag_in_your_hand1

所给页面提交数据会有反馈↔



Token: 1

Get flag!

Wrong!

xMpCOKC5I4INzFCab3WEmw

此时在所给 js 文件中找到提交数据所经过的函数

```
7 L}
   Efunction ck(s) {
8
9
         try {
0
             ic
        } catch (e) {
1
2
             return;
3
        }
        var a = [118, 104, 102, 120, 117, 108, 119, 124, 48, 123, 101, 120];
4
5
        if (s.length == a.length) {
             for (i = 0; i < s.length; i++) {</pre>
6
7
                 if (a[i] - s.charCodeAt(i) != 3)
                     return ic = false;
8
9
             }
0
             return ic = true;
1
         }
2
        return ic = false;
3
   L
```

说明字符串应为 a 中数字减 3, python 脚本为:

```
<u>File Edit Format Run Options Window Help</u>
d a =[115, 101, 99, 117, 114, 105, 116, 121, 45, 120, 98, 117]
s =
for i in a:
   s += chr(i)
 print(s)
security-xbu
```

Flag in your Hand

Type in some token to get the flag.

Tips: Flag is in your hand.

Token: security-xbu

Get flag!

You got the flag below!!

RenIbyd8Fgg5hawvQm7TDQ

https://blog.csdn.net/weixin_44159598

safer-than-rot13

cry100

XMVZGC RGC AMG RVMG HGFGMQYCD VT VWM BYNO, NSVWDS NSGO RAO XG UWFN AF HACDGMVWF. AIRVFN AII AMG JVRRVC-XVMC, FYRBIG TVIZ ESV SAH CGQGM XGGC RVMG NSAC A RYIG TMVR NSG SVWFG ESGMG NSGO EGMG XVMC WCNYI NSG HAO FVRG IVMH JARG MVWCH NV NAZG NSGR VTT NV EAM. OVWM TIAD YF "CV NSYF YF CVN JMOBNV RO HGAM", YC IVEGMJAFG, EYNS WCHGMFJVMGF YCFNGAH VT FBAJGF, FWMMVWCHGH XO NSG WFWAI "TIAD" NAD ACH JWMIO XMAJGF. GCUVO.

文字大批量,所以进行词频分析↔

0	-1.258	BROKEN MEN ARE MORE DESERVING OF OUR PITY, THOUGH THEY MAY BE JUST AS DANGEROUS. ALMOST ALL ARE COMMON-BORN, SIMPLE FOLK WHO HAD NEVER BEEN MORE THAN A MILE FROM THE HOUSE WHERE THEY WERE BORN UNTIL THE DAY SOME LORD CAME ROUND TO TAKE THEM OFF TO WAR. YOUR FLAG IS "NO THIS IS NOT CRYPTO MY DEAR", IN LOWERCASE, WITH UNDERSCORES INSTEAD OF SPACES, SURROUNDED BY THE USUAL "FLAG" TAG AND CURLY BROKES. EXJOL
1	-1.542	BROKEN MEN ARE MORE DESERVING OF OUR PITY, THOUGH THEY MAY BE JUST AS DANGEROUS. ALMOST ALL ARE COMMON- BORN, SIMPLE FOLK WHO HAD NEVER BEEN MORE THAN A MILE FROM THE HOUSE WHERE THEY WERE BORN UNTIL THE DAY SOME LORD CAME ROUND TO TAKE THEM OFF TO WAR. YOUR FLAG IS NO THIS IS NOT CRYPTOMY DEAR, IN LOWER CASE, WITH UNDERSCORES INSTEAD OF SPACES, SURROUNDED BY THE USUAL FLAG TAG AND CURLY BRACES. ENJOY.
2	-3. 128	CRAVEL DEL URE DARE FENERGOLK AS AIR ZOTY, THAIKH THEY DUY CE PINT UN FULKERAIN. UMDANT UMM URE BADDAL-CARL, NODZME SAMV WHA HUF LEGER CEEL DARE THIL U DOME SRAD THE HAINE WHERE THEY WERE CARL ILTOM THE FLY NADE MARF BUDE RAILF TA TUVE THED ASS TA WUR. YAIR SMUK ON "LA THON ON LAT BRYZTA DY FEUR", OL MAWERBUNE, WOTH ILFERNBAREN OLNTEUF AS NZUBEN, NIRRAILFEF CY THE INIUM "SMUK" TUK ULF BIRMY CRUBEN. ELPAY.
3	-3. 312	BLAXEN MEN OLE MALE DEKELVING AT AUL WIPS, PRAUGR PRES MOS BE YUKP OK DONGELAUK. OZMAKP OZZ OLE HAMMAN-BALN, KIMWZE TAZX CRA ROD NEVEL BEEN MALE PRON O MIZE TLAM PRE RAUKE CRELE PRES CELE BALN UNPIZ PRE DOS KAME ZALD HOME LAUND PA POXE PREM ATT PA COL. SAUL TZOG IK "NA PRIK IK NAP HLSWPA MS DEOL", IN ZACELHOKE, CIPR UNDELKHALEK INKPEOD AT KWOHEK, KULLAUNDED BS PRE UKUOZ "TZOG" POG OND HULZS BLOHEK. ENYAS.
4	-3.320	HRAVED MED ORE MARE BENERGUDP AS AIR ZUCY, CLAIPL CLEY MOY HE WINC ON BODPERAIN. OTMANC OTT ORE KAMMAD-HARD, NUMZTE SATV FLA LOB DEGER HEED MARE CLOD O MUTE SRAM CLE LAINE FLERE CLEY FERE HARD IDCUT CLE BOY NAME TARB KOME RAIDB CA COVE CLEM ASS CA FOR. YAIR STOP UN "DA CLUN UN DAC KRYZCA MY BEOR", UD TAFERKONE, FUCL IDBERNKAREN UDNCEOB AS NZOKEN, NIRRAIDBEB HY CLE INIOT "STOP" COP ODB KIRTY HROKEN. EDWAY.
5	-3, 355	GNEJAS BAS INA BENA MARANZOSC EV EUN FOTY, THEUCH THAY BIY GA PURT IR MISCANEUR. ILBERT ILL INA DEBBES-GENS, ROBFLA VELJ KHE HIM SAZAN GAAS BENA THIS I BOLA VNEB THA HEURA KHANA THAY KANA GENS USTOL THA MIY REBA LENM DIBA NEUSM TE TIJA THAB EVV TE KIN. YEUN VLIC OR "SE THOR OR SET DNYFTE BY MAIN", OS LEKANDIRA, KOTH USMANRDENAR OSRTAIM EV RFIDAR, RUNNEUSMAM GY THA URUIL "VLIC" TIC ISM DUNLY GNIDAR. ASPEY.
6	-3, 395	BNOMAD KAD INA KONA SATANGEDP OF OLN UECY, CHOLPH CHAY KIY BA VLTC IT SIDPANOLT. IRKOTC IRR INA JOKKOD-BOND, TEKURA FORM WHO HIS DAGAN BAAD KONA CHID I KERA FNOK CHA HOLTA WHANA CHAY WANA BOND LDCER CHA SIY TOKA RONS JIKA NOLDS CO CIMA CHAK OFF CO WIN, YOLN FRIP ET "DO CHET ET DOC JNYUCO KY SAIN", ED ROWANJITA, WECH LDSANTJONAT EDICAIS OF JUIJAT, TLNNOLISAS BO CHA LITLR "P2IP" CIP IDS JLNRY BNIJAT. ADVOY.

• 提交flag,将空格替换为下划线""

Decode_The_File

- > Ipercent for the for th
- 4 IyBBdXRob3I6ICAgVG9k2CBXaG10ZW1hbk==
- 5 IyBEYXR101AgICAgN3RoIE1heSwgHjAwH/--
- 6 IyBWZXJpb246ICAgMS4x
- 7 IyBIb211cGFnZTogaHR0cDovL2hvbWUucGFjaWZpYy5uZXQuYXUvfnR3aG10ZW1hL2R1cy5odG1s
- S I0---
- 9 IyBNb2RpZmljYXRpb25zIHRvIDNkZXMgQ0JDIGNvZGUgYnkgTWF0dCBKb2huc3Rvb1AyMDA0IDxtYXR0IGF0IHVjYyBhc24gYXU+
- 10 15=-
- I IyBUaG1zIGFs229yaXRobSBpcyBhIHB1cmUgcH10aG9uIG1tcGx1bWVudGF0aW9uIG9mIHRoZSBERVMgYWxnb3JpdGhtL1==
- 12 Iy8JdC8pcy8pbiBwdXJ1IH85dGhvbiB0by8hdm9pZC8wb3J0YWJpbG10e58pc3N1ZXM#IHNpbmN1IG1vc3QgREVTID==
- I JSDXBsZW11bnRhdG1vbnMgYXJ1IHByb2dyYW1tZWQgaW4gQyAoZm9yIHB1cmZvcm1hbmN1IHJ1YXNvbnMpLn== 14 I7==
- -----
 - IyBUcmlwbGUgREVTIGNsYXNzIG1zIGFsc28gaWiwbGVt2W50ZWQsIHV0aWxpc21uZyB0aGUgREVTIGJhc2UuIFRyaXBsZSBERVN=
- IyBpcyBlaXRoZXIgREVTLUVERTMgd210aCBhIDI0IGJ5dGUga2V5LCBvciBERVMtRURFMiB3aXRoIGEgMTYgYn102SBrZXku I I4==
- 18 IyBTZWUgdGhlIFJFQURNRS50eHQgdGhhdCBzaG91bGQgY29tZSB3aXRoIHRoaXMgcHl0aG9uIG1vZHVsZSBmb3IgdGhl
- 19 IyBpbXBsZW11bnRhdGlvbiBtZXRob2RzIHVzZWQu
- IIIIQSBwdXJIIHB5dGhvbiBpbXBsZWIlbnRhdGlvbiBvZiB0aGUgREVTIGFuZCBUUklQTEUgRZVTIGVuY3J5cHRpb24gYwxnb3JpdGhtc5=

• 显然,这里的数据是通过 Base64 算法编码的,让我们对其进行解码:

Documentation + 3 4 # Author: Todd Whiteman 5 7th May, 2003 # Date: # Verion: 1.1 6 7 # Homepage: http://home.pacific.net.au/~twhitema/des.html 8 9 # Modifications to 3des CBC code by Matt Johnston 2004 <matt at ucc asn au> # This algorithm is a pure python implementation of the DES algorithm. 12 # It is in pure python to avoid portability issues, since most DES # implementations are programmed in C (for performance reasons). 14 15 # Triple DES class is also implemented, utilising the DES base. Triple DES # is either DES-EDE3 with a 24 byte key, or DES-EDE2 with a 16 byte key. 16 17 18 # See the README.txt that should come with this python module for the # implementation methods used. 19

 https://github.com/n0fate/chainbreaker/blob/master/pyDes.py,将解码后的文件与从上述链接下载的文件进行了比较,我们找 不到添加到解码后文件的任何其他信息。因此,必须在 Base64字符串中隐藏一些秘密,但是它们是什么?让我们使用标准 的 Base64算法对内容进行重新编码,以查看是否存在任何差异:



 如果你熟悉 Base64,我认为你可以轻松找出原因。众所周知,对于 Base64算法,原始数据将被分成 3个字节的组,如果 最后一组仅包含 1个或 2个字节,它将在末尾添加一些填充并使用 1个或 2个"="表示最后一组中有多少个原始字节。这是最 后一组中 1个字节的示例:

Text content		м										nu	II				null								
ASCII	77 (0x4d)									0 (0x00)							0 (0x00)								
Bit pattern	0	1	0	0	1	1	0	1	0	0	0	0		-	-	-	-	-	-	-	-	-	-	-	
Index		19							2	22		5	null						null						
Base64-encoded		т							w								die	=							
														4	pa	100		igs							

实际上,此处的4个填充将被解码例程忽略,也就是说,我们可以在此处放置任何位,这是隐藏信息的好地方!理解这一点,解决这个挑战将没有困难,以下脚本是我用来提取隐藏信息的工具:

```
import base64
                                                                 🕀 🖹 E
import string
def tobin(data):
   b64table = string.ascii_uppercase + string.ascii_lowercase + string.digits +
1+/1
   index = b64table.find(data)
   return format (index, '06b')
def toStr(bin):
   binlen = len(bin)
   out = ''
   for i in range(0, binlen, 8):
       out += chr(int(bin[i:i+8], 2))
   return out
out = ''
for line in open('cip_d0283b2c5b4b87423e350f8640a0001e', 'rb'):
   line = line.strip()
   if line.strip() [-2:] == '==':
       binstr = tobin(line[-3:-2])
       out += binstr[-4:]
       print binstr[-4:]
    elif line.strip()[-1:] == '=':
      binstr = tobin(line[-2:-1])
       out += binstr[-2:]
       print binstr[-2:]
print out
print toStr(out)
```

• Flag: ROIS{base_GA_caN_b3_d1ffeR3nT}