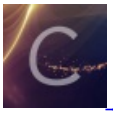
base64原理+base64隐写

原创

大千SS 于 2019-04-14 16:56:58 发布 6203 收藏 35
分类专栏： 隐写、杂项 密码知识 文章标签： base64 base64隐写
版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/zz_Caleb/article/details/89298335
版权


隐写、杂项 同时被 2 个专栏收录
24 篇文章 1 订阅
订阅专栏

密码知识
32 篇文章 1 订阅
订阅专栏

借鉴一位大佬的博客，说一下base64原理及base64隐写的原理：

转载自大佬博客：https://www.tr0y.wang/2017/06/14/Base64steg/

BASE64 是一种编码方式, 是一种可逆的编码方式.
编码后的数据是一个字符串, 包含的字符为: A-Za-z0-9+/
共 64 个字符：26 + 26 + 10 + 1 + 1 = 64
其实是 65 个字符, “=”是填充字符.

64 个字符需要 6 位二进制来表示, 表示成数值为 0~63.

Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

图片来自：www.tr0y.wang

这样, 长度为 3 个字节的数据经过 Base64 编码后就变为 4 个字节

编码

比如, 字符串”Tr0”经过 Base64 编码后变为”VHlw”

T								r								0							
0	1	0	1	0	1	0	0	0	1	1	1	0	0	1	0	0	0	1	1	0	0	0	0

0	1	0	1	0	1	0	0	0	1	1	1	0	0	1	0	0	0	1	1	0	0	0	0
V						H						I						w					

图片来自：www.troy.wang

上面说的字符串长度为 3 个字节的数据位数是 8x3=24, 可以精确地分成 6x4.
如果字节数不是 3 的倍数, 则位数就不是 6 的倍数, 那么就不能精确地划分成 6 位的块.
此时, 需在原数据二进制值后面添加零, 使其字节数是 6 的倍数.
然后, 在编码后的字符串后面添加 1 个或 2 个等号"=", 表示所添加的零值字节数.
比如, 字符串"Tr0y"经过 Base64 编码后变为"VHlweQ=="

T								r								0								y																							
0	1	0	1	0	1	0	0	0	1	1	1	0	0	1	0	0	0	1	1	0	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V						H						I						w						e						Q						=						=					

图片来自：www.troy.wang

橙色底纹就是添加的 0.
这是 Base64 编码的方式.

解码

解码就是编码的逆过程.

1. 把 Base64 字符串去掉等号, 转为二进制数(VHlweQ== -> VHlweQ -> 010101000111001000110000011110010000).
2. 从左到右, 8 个位一组, 多余位的扔掉, 转为对应的 ASCII 码(01010100 01110010 00110000 01111001 0000 -> 扔掉最后 4 位 -> 01010100 01110010 00110000 01111001 -> Tr0y)

隐写原理

注意红色的 0, 我们在解码的时候将其丢弃了, 所以这里的值不会影响解码. 所以我们可以在这进行隐写.
为什么等号的那部分 0 不能用于隐写? 因为修改那里的二进制值会导致等号数量变化, 解码的第 1 步会受影响. 自然也就破坏了源字符串.
而红色部分的 0 是作为最后一个字符二进制的组成部分, 还原时只用到了最后一个字符二进制的前部分, 后面的部分就不会影响还原.
唯一的影响就是最后一个字符会变化. 如下图

T								r								0								y																							
0	1	0	1	0	1	0	0	0	1	1	1	0	0	1	0	0	0	1	1	0	0	0	0	0	1	1	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
V						H						I						w						e						R						=						=					

图片来自：www.troy.wang

如果你直接解密'VHlweQ=='与'VHlweR==', 得到的结果都是'Tr0y'.

当然, 一行 base64 顶多能有 2 个等号, 也就是有 2*2 位的可隐写位. 所以我们得弄很多行, 才能隐藏一个字符串, 这也是为什么题目给了一大段 base64 的原因.
接下来, 把要隐藏的 flag 转为 8 位二进制, 塞进去就行了.

以上就是隐写的原理了，这里以一个题目为例。

题目给了一个加密了的压缩包，伪加密破解得到许多的base64码：

```
U3RlZ2Fub2dyYXBoeSBpcyB0aGUgYXJ0IGFuZCBzY2llbmNlIG9m
IHdyaXRpbmcgaGlkZGVuIG1lc3NhZ2VzIGluIHN1Y2ggYSB3YXkgdGhhdCBubyBvbmV=
LCBhcGFydCBmcm9tIHRoZSBzZW5kZXIgYW5kIGludGVuZGVkIHJlY2lwaWVudCwgc3VzcGU=
Y3RzIHRoZSBleGlzdGVuY2Ugb2YgdGhlIG1lc3M=
YWdlLCBhIGZvcm0gb2Ygc2VjdXJpdHkgdGhyb3VnaCBvYnNjdXJpdHkuIFS=
aGUgd29yZCBzdGVnYW5vZ3JhcGh5IGlzIG9mIEdyZWVrIG9yaWdpbiBhbmQgbWVhbnMgImNvbmNlYW==
```

bGVkIHdyaXRpbmciIGZyb20gdGhlIEdyZWVrIHdvcmRzIHN0ZWdhbm9zIG1lYW5pbmcgImNv
dmVyZWQgb3IgcHJvdGVjdGVkIiwgYW5kIGdyYXBoZWluIG1lYW5pbmcgInRvIHc=
cml0ZSIuIFRoZSBmaXJzdCByZWNvcmRlZCB1c2Ugb2YgdGhlIHRlcm0gd2FzIGluIDE0OTkgYnkgSm9o
YW5uZXMgVHJpdGhlbWl1cyBpbiBoaXMgU3RlZ2Fub2dyYXBoaWEsIGEgdHJlYV==
dGlzZSBvbiBjcnlwdG9ncmFwaHkgYW5kIHN0ZWdhbm9ncmFwaHkgZGlzZ8==
dWlzZWQgYXMgYSBib29rIG9uIG1hZ2ljLiBHZW5lcmFsbHksIG1lc3P=
YWdlcyB3aWxsIGFwcGVhciB0byBiZSBzb21ldGhpbmcgZWxzZTogaW1hZ2VzLCBhcnRp
Y2xlcywgc2hvcHBpbmcgbGlzdHMsIG9yIHNvbWUgb3R=
aGVyIGNvdmVydGV4dCBhbmQsIGNsYXNzaWNhbGx5LCB0aGUgaGlkZGVuIG1lc3NhZ2UgbWF5IGJlIGluIGludmm=
c2libGUgaW5rIGJldHdlZW4gdGhlIHZpc2libGUgbGluZXMgb2YgYSBwcml2YXRlIGxldHRlci4NCg0KVGhl
IGFkdmFudGFnZSBvZiBzdGVnYW5vZ3JhcGh5LCBvdmVyIGNy
eXB0b2dyYXBoeSBhbG9uZSwgaXMgdGhhdCBtZXNzYWdlcyBkbyBub3QgYXR0cmFjdCBhdHRlbnRpb25=
IHRvIHRoZW1zZWx2ZXMuIFBsYWlubHkgdmlzaWJsZSBlbmNyeXB0ZWQgbWVzc2FnZXOXbm8gbWF0dGVyIF==
aG93IHVuYnJlYWthYmxll3dpbGwgYXJvdXNlIHN=
dXNwaWNpb24sIGFuZCBtYXkgaW4gdGhlbXNlbHZlcyBiZSBpbmNyaW1pbmF0aW5nIP==
aW4gY291bnRyaWVzIHdoZXJlIGVuY3J5cHRpb24gaXMgaWxsZWdhbC4gVGhlcmVmb3JlLH==
IHdoZXJlYXMgY3J5cHRvZ3JhcGh5IHByb3RlY3RzIHRoZSBjb250ZW50cyBvZj==
IGEgbWVzc2FnZSwgc3RlZ2Fub2dyYXBoeSBjYW4gYmUgc2FpZCB0byBwcm90ZWN0IGJ=
b3RoIG1lc3NhZ2VzIGFuZCBjb21tdW5pY2F0aW5nIHBhcnRpZXMuDQoNClN0ZWdhbm9ncmFwaHkgaW5jbHW=
ZGVzIHRoZSBjb25jZWFsbWVudCBvZiBpbmZvcm1hdGlvbiB3aXRoaW4gY29t
cHV0ZXIgZmlsZXMuIEluIGRpZ2l0YWwgc3RlZ2Fub2dyYXBoeSwgZWxlY3Ryb25pYyBjb21tdW5pY2F0aW9u
cyBtYXkgaW5jbHVkZSBzdGVnYW5vZ3JhcGhpYyBjb2Rpbmcgaw5zaZ==
ZGUgb2YgYSB0cmFuc3BvcnQgbGF5ZXIsIHN1Y2ggYXMgYSBkb2N1bWVudCBmaWxlLCBpbWFnZSBmaWx=
ZSwgcHJvZ3JhbSBvciBwcm90b2NvbC4gTWVkaWEg
ZmlsZXMgYXJlIGlkZWFsIGZvciBzdGVnYW5vZ3JhcGhpYyB0cmFuc21pc3Npb+==
biBiZWNhdXNlIG9mIHRoZWlyIGxhcmdlIHNpemUuIEFzIB==
YSBzaW1wbGUgZXhhbXBsZSwgYSBzZW5kZXIgbWlnaHQgc3RhcnQgd2l0aCBh
biBpbm5vY3VvdXMgaW1hZ2UgZmlsZSBhbmQgYWRqdXN0IHRoZSBjb2xvciBvZiBldmVyeSAxMDB0aCBwaXhlbCD=
dG8gY29ycmVzcG9uZCB0byBhIGxldHRlciBpbiB0aGUgYWxwaGFiZXQsIGF=
IGNoYW5nZSBzbyBzdWJ0bGUgdGhhdCBzb21lb25lIG5vdCBzcGVjaWZpY2FsbHkgbG9va2luZyBm
b3IgaXQgaXMgdW5saWtlbHkgdG8gbm90aWNlIGl0Lg0KDQpUaGU=
IGZpcnN0IHJlY29yZGVkIHVzZXMgb2Ygc3RlZ2Fub2dyYXBoeSBjYW4gYmUgdHJ=
YWNlZCBiYWNrIHRvIDQ0MCBCQyB3aGVuIEhlcm9kb3R1cyBtZW50aW9ucyB0d28gZXhhbXBsZXMgb+==
ZiBzdGVnYW5vZ3JhcGh5IGluIFRoZSBIaXN0b3JpZXMgb2Yg
SGVyb2RvdHVzLiBEZW1hcmF0dXMgc2VudCBhIHdhcm5pbmcgYWJvdXQgYSB=
Zm9ydGhjb21pbmcgYXR0YWNrIHRvIEdyZWVjZSBieSB3
cml0aW5nIGl0IGRpcmVjdGx5IG9uIHRoZSB3b29kZW4gYmFja2luZyBvZiBhIHdheCB0YWJsZXQgYmVm
b3JlIGFwcGx5aW5nIGl0cyBiZWVzd2F4IHN1cmZhY2UuIFdheCB0YWJsZXRzIHdlcmUgaW4gY29tbW9uIHVzZV==
IHRoZW4gYXMgcmV1c2FibGUgd3JpdGluZyBzdXJmYWNlcywgc29tZXRpbWX=
cyB1c2VkIGZviBzaG9ydGhhbmQuIEFub3RoZXIgYW5jaWVudCBleGFtcGxlIGlzIHRoYXQgb9==
ZiBIaXN0aWFldXMsIHdobyBzaGF2ZWQgdGhlIGhlYWQgb2YgaGlzIG1vc3QgdHJ1c3RlZCBz
bGF2ZSBhbmQgdGF0dG9vZWQgYSBtZXNzYWdlIG9uIGl0LiBBZnRlciBoaXMgaGFpciBoYWQgZ5==
cm93biB0aGUgbWVzc2FnZSB3YXMgaGlkZGVuLiBUaGUgcHVycG9zZSB3YXMgdG+=
IGluc3RpZ2F0ZSBhIHJldm9sdCBhZ2FpbnN0IHRoZSBQZXJzaWFucy4NCg0KU3RlZ2Fub2dyYXBoeSBoYXMgYm==
ZWVuIHdpZGVseSB1c2VkLCBpbmNsdWRpbmcgaW4gcmVjZW50IGhpc3RvcmljYWwgdGltZXMgYW5kIHT=
aGUgcHJlc2VudCBkYXkuIFBvc3NpYmxlIHBlcm11dGF0aW9ucyBhcmUgZW5kbGVzcyBhbmT=
IGtub3duIGV4YW1wbGVzIGluY2x1ZGU6DQoqIEhpZGRlbiBtZXNzYWdlcyB3aXRoaW4gd2F4IHRh
YmxldHM6IGluIGFuY2llbnQgR3JlZWNlLCBwZW9wbGUgd3JvdGUgbWV=
c3NhZ2VzIG9uIHRoZSB3b29kLCB0aGVuIGNvdmVyZWQgaXQgd2l0aCB3YXggdXBvbiB3aGljaCBhbiBpbm5vY2Vu
dCBjb3ZlcmluZyBtZXNzYWdlIHdhcyB3cml0dGVu
Lg0KKiBIaWRkZW4gbWVzc2FnZXMgb24gbWVzc2VuZ2VyJ3MgYm9keTogYWxzbyB1c2VkIGluIGFuY2llbt==
dCBHcmVlY2UuIEhlcm9kb3R1cyB0ZWxscyB0aGUgc3Rvcnkgb1==
ZiBhIG1lc3NhZ2UgdGF0dG9vZWQgb24gYSBzbGF2ZSdzIHNoYXZlZCBoZWFkLCBoaWRkZW4gYnkgdGhl
IGdyb3d0aCBvZiBoaXMgaGFpciwgYW5kIGV4cG9zZWQgYnkgc2hhdmluZyBoaXMgaGVhZM==
IGFnYWluLiBUaGUgbWVzc2FnZSBhbGxlZ2VkbHkgY2FycmllZCBhIHdhcm5pbmcgdG8gR3JlZWNlIGFib5==
dXQgUGVyc2lhbiBpbnZhc2lvbiBwbGFucy4gVGh=
aXMgbWV0aG9kIGhhcyBvYnZpb3VzIGRyYXdiYWNrcyz=
IHN1Y2ggYXMgZGVsYXllZCB0cmFuc21pc3Npb24gd2hpbGUgd2FpdGluZyBmb3IgdGhlIHP=
bGF2ZSdzIGhhaXIgdG8gZ3JvdywgYW5kIHRoZSByZXN0cmljdGlvbnMgb3==
biB0aGUgbnVtYmVyIGFuZCBzaXplIG9mIG1lc3M=

```
YWdlcyB0aGF0IGNhbiBiZSBlbmNvZGVkIG9uIG9uZSBwZXJzb24=
J3Mgc2NhbHAuDQoqIEluIFdXSUksIHRoZSBGcmVuY2ggUmVzaXN0YW5jZSBzZW50IHNvbWUgbWVzc2FnZXMgd2==
cml0dGVuIG9uIHRoZSBiYWNrcyBvZiBjb3VyaWVycyD=
dXNpbmcgaW52aXNpYmxlIGluay4NCiogSGlkZGVuIG1lc3NhZ2VzIG9uIHBhcGVyIHdy
aXR0ZW4gaW4gc2VjcmV0IGlua3MsIHVuZGVyIG90aGVyIG1lc3NhZ2Vz
IG9yIG9uIHRoZSBibGFuayBwYXJ0cyBvZiBvdGhlct==
IG1lc3NhZ2VzLg0KKiBNZXNzYWdlcyB3cml0dGVuIGluIE1vcnNlIGNvZGUgb24ga25pdHRpbmcgeWFybiBhbmQg
dGhlbiBrbml0dGVkIGludG8gYSBwaWVjZSBvZiBjbG90aGluZyB3b3K=
biBieSBhIGNvdXJpZXIuDQoqIE1lc3NhZ2VzIHdyaXR0ZW4gb24gdGhlIGJhY2sgb5==
ZiBwb3N0YWdlIHN0YW1wcy4NCiogRHVyaW5nIGFuZCBhZnRlcm==
IFdvcmxkIFdhciBJSSwgZXNwaW9uYWdlIGFnZW50cyB1c2VkIHBob3RvZ3JhcGhpY2FsbHkgcO==
cm9kdWNlZCBtaWNyb2RvdHMgdG8gc2VuZCBpbmZvcm1hdGlvbiBiYWNrIGFuZH==
IGZvcnRoLiBNaWNyb2RvdHMgd2VyZSB0eXBpY2FsbHkg
bWludXRlLCBhcHByb3hpbWF0ZWx5IGxlc3MgdGhhbiB0aGUgc2l6ZSBvZiB0aGUgcGVyaW9kIHByb2R=
dWNlZCBieSBhIHR5cGV3cml0ZXIuIFdXSUkgbWljcm9kb3RzIG5lZWRlZCB0byBiZSBlbWJlZGRlZB==
IGluIHRoZSBwYXBlciBhbmQgY292ZXJlZCB3aXRoIGFuIGFkaGVzaXZlIChzdWNoIGFzIGNvbGxvZGlvbikuIFR=
aGlzIHdhcyByZWZsZWN0aXZlIGFuZCB0aHVzIGRldGVjdGFibGUg
Ynkgdmlld2luZyBhZ2FpbnN0IGdsYW5jaW5nIGxpZ2h0LiBBbHRlcm5hdGl2ZSB0ZWNobmlxdWVzIGluY2x1ZGVk
IGluc2VydGluZyBtaWNyb2RvdHMgaW50byBzbGl0cyBjdXQgaW50byB0aGUgZWRnZSBvZv==
IHBvc3QgY2FyZHMuDQoqIER1cmluZyBXb3JsZCBXYXIgSUksIGEgc3B5IGZvciB=
SmFwYW4gaW4gTmV3IFlvcmsgQ2l0eSwgVmVsdmFsZWW=
IERpY2tpbnNvbiwgc2VudCBpbmZvcm1hdGlvbiB0byBhY2NvbW1vZGF0aW9=
biBhZGRyZXNzZXMgaW4gbmV1dHJhbCBTb3V0aCBBbWVyaWO=
YS4gU2hlIHdhcyBhIGRlYWxlciBpbiBkb2xscywgYW5kIG==
aGVyIGxldHRlcnMgZGlzY3Vzc2VkIGhvdyBtYW55IG9mIHRoaXMgb3IgdGhhdCBkb2xs
IHRvIHNoaXAuIFRoZSBzdGVnb3RleHQgd2FzIHRoZSBkb2xsIG9yZGVycywgd2hpbGUgdGhl
IGNvbmNlYWxlZCAicGxhaW50ZXh0IiB3YXMgaXRzZWxmIGVuY2+=
ZGVkIGFuZCBnYXZlIGluZm9ybWF0aW9uIGFib3V0IHNoaXAgbW92ZW1lbnRzLF==
IGV0Yy4gSGVyIGNhc2UgYmVjYW1lIHNvbWV3aGF0IGZh
bW91cyBhbmQgc2hlIGJlY2FtZSBrbm93biBhcyB0aGX=
IERvbGwgV29tYW4uDQoqIENvbGQgV2FyIGNvdW50
ZXItcHJvcGFnYW5kYS4gSW4gMTk2OCwgY3JldyBtZW1iZW==
cnMgb2YgdGhlIFVTUyBQdWVibG8gKEFHRVItMikgaW50ZWxsaWdlbmNlIHNoaXAgaGVsZCBhcyBwcm==
aXNvbmVycyBieSBOb3J0aCBLb3JlYSwgY29tbXVuaWNhdGVkIGluIHNpZ25=
IGxhbmd1YWdlIGR1cmluZyBzdGFnZWQgcGhvdG8gb3Bwb3J0
dW5pdGllcywgaW5mb3JtaW5nIHRoZSBVbml0ZWQgU3RhdGVzIHRoZXkg
d2VyZSBub3QgZGVmZWN0b3JzIGJ1dCByYXRoZXIgd2VyZSBiZWluZyBoZWxkIGNh
cHRpdmUgYnkgdGhlIE5vcnRoIEtvcmVhbnMuIEluIG90aGVyIHBob3Rv
cyBwcmVzZW50ZWQgdG8gdGhlIFVTLCBjcmV3IG1lbWJlcnMgZ2F2ZSAidGhlIGZpbmdlciIgdG8g
dGhlIHVuc3VzcGVjdGluZyBOb3J0aCBLb3JlYW5zLCBpbiBhbiBhdHRlbXB0IHRvIE==
ZGlzY3JlZGl0IHBob3RvcyB0aGF0IHNob3dlZCB0aGVtIHNtaQ==
bGluZyBhbmQgY29tZm9ydGFibGUuDQoNCi0tDQpodHRwOi8vZW4ud2lraXBlZGlhLm9yZw==
L3dpa2kvU3RlZ2Fub2dyYXBoeQ0K
```


我们要做的就是解决这个base64隐写，解密脚本如下：

```
import base64

b64chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
with open('stego.txt', 'rb') as f:
    flag = ''
    bin_str = ''
    for line in f.readlines():
        stegb64 = str(line, "utf-8").strip("\n")
        rowb64 =  str(base64.b64encode(base64.b64decode(stegb64)), "utf-8").strip("\n")
        offset = abs(b64chars.index(stegb64.replace('=','')[-1]) - b64chars.index(rowb64.replace('=','')[-1
        equalnum = stegb64.count('=') #no equalnum no offset
        if equalnum:
            bin_str += bin(offset)[2:].zfill(equalnum * 2)
            #flag += chr(int(bin(offset)[2:].zfill(equalnum * 2), 2))
            #print(flag) 这样写得不出正确结果
        print([chr(int(bin_str[i:i + 8], 2)) for i in range(0, len(bin_str), 8)])
```

顺带给出加密脚本：

```
转载自：https://www.tr0y.wang/2017/06/14/Base64steg/
import base64
flag = 'Tr0y{Base64isF4n}' #flag
bin_str = ''.join([bin(ord(c)).replace('0b', '').zfill(8) for c in flag])
base64chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
with open('0.txt', 'rb') as f0, open('1.txt', 'wb') as f1: #'0.txt'是明文, '1.txt'用于存放隐写后的 base64
    for line in f0.readlines():
        rowstr = base64.b64encode(line.replace('\n', ''))
        equalnum = rowstr.count('=')
        if equalnum and len(bin_str):
            offset = int('0b'+bin_str[:equalnum * 2], 2)
            char = rowstr[len(rowstr) - equalnum - 1]
            rowstr = rowstr.replace(char, base64chars[base64chars.index(char) + offset])
            bin_str = bin_str[equalnum*2:]
        f1.write(rowstr + '\n')
```