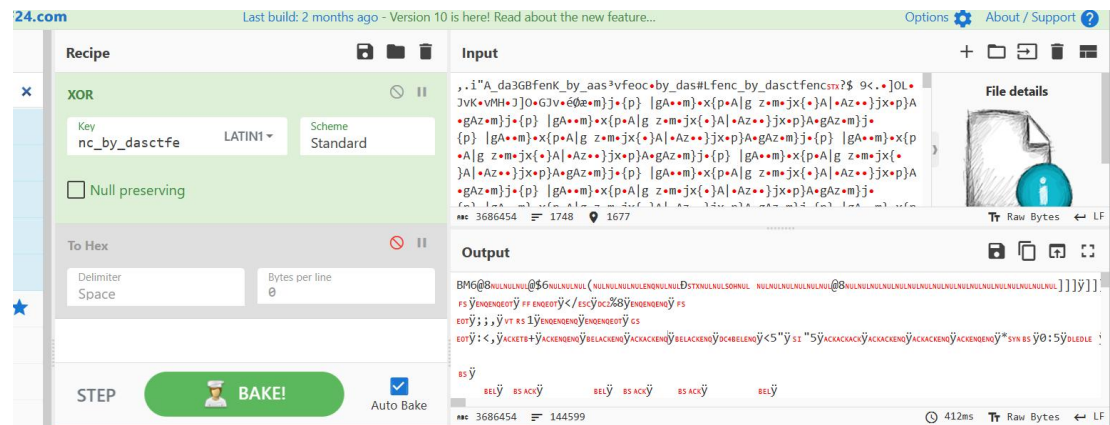


Cap:

将 cap.bin 文件和 nc_by_dasctfe 异或，得到一张图片



Ez_exe:

用 pyinstxtractor.py 解包

接着用 pycdc 将 ez_py.pyc 还原成 py 文件

一开始我的 pycdc 也转不了，后来按照 Gift1a 师傅的说法，改一下文件头

| 起始页 | encrypted.bin | cap.bin | ez_py.pyc x |
|--------|---------------------------------|---------------------------------|---------------------------------|
| | 0 1 2 3 4 5 6 7 8 9 A B C D E F | 0 1 2 3 4 5 6 7 8 9 A B C D E F | 0 1 2 3 4 5 6 7 8 9 A B C D E F |
| 0000h: | A7 0D 0D 0A 03 00 00 00 | 00 00 00 00 00 00 00 00 | \$..... |
| 0010h: | E3 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00 | ä..... |
| 0020h: | 00 00 00 00 00 F3 DC 03 | 00 00 97 00 64 00 64 01 |öÜ...-d.d. |
| 0030h: | 6C 00 5A 00 64 00 64 02 | 6C 01 54 00 64 00 64 02 | l.Z.d.d.l.T.d.d. |

就可以还原了，得到如下代码

```
33 def __init__(self, process_id, c_ulong):
34     self.process_id = process_id
35     self.c_ulong = c_ulong
36
37 StartupInfo = _STARTUPINFO()
38 ProcessInfo = _PROCESS_INFORMATION()
39 key1 = bytes(md5(b'bin1bin1bin1').hexdigest().encode())
40 file = open('bin1', 'rb').read()
41 arr = range(len(file))()
42 open('bin1', 'wb').write(bytes(arr))
43 sleep(0)
44 bet = ctypes.windll.kernel32.CreateProcessA(b'bin1', ctypes.c_int(0), ctypes.c_int(0), ctypes.c_int(0), ctypes.c_int(0), ctypes.c_int(0), ctypes.c_int(0), ctypes.c_int(0), ctypes.c_int(0), ctypes.c_int(0))
45 open('bin1', 'wb').write(file)
```

将 bin1bin1bin1 字符串 md5 加密得到 ff81074ce54dbc3b394fe8821e909706



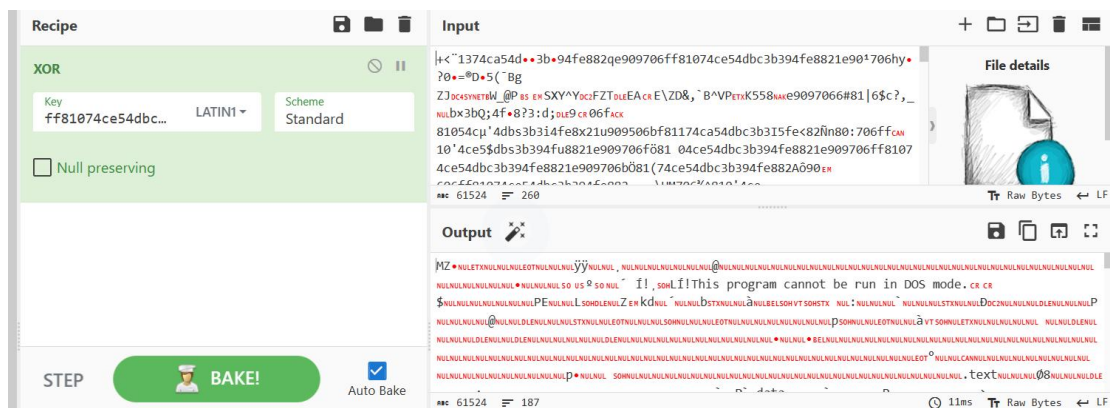
MD5在线加密

要加密的字符串: bin1bin1bin1

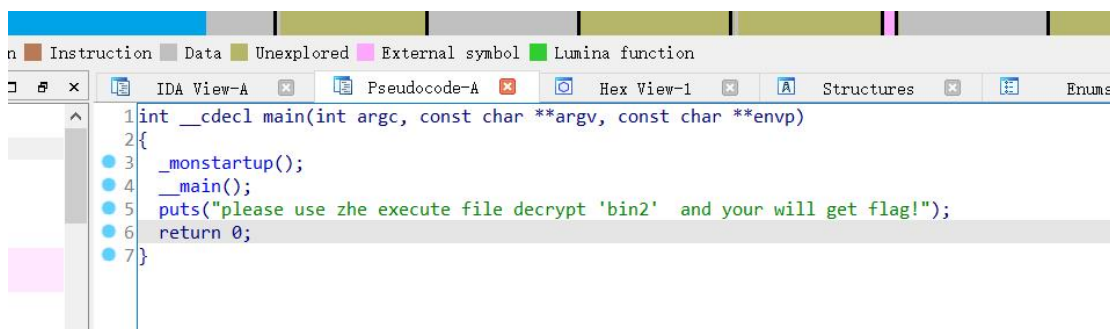
加密

| | |
|--------|----------------------------------|
| 字符串 | bin1bin1bin1 |
| 16位 小写 | e54dbc3b394fe882 |
| 16位 大写 | E54DBC3B394FE882 |
| 32位 小写 | ff81074ce54dbc3b394fe8821e909706 |
| 32位 大写 | FF81074CE54DBC3B394FE8821E909706 |

接着将 bin1 和 ff81074ce54dbc3b394fe8821e909706 异或一下
得到一个 exe 文件



IDA 看一下，提示用得到的程序解密 bin2 文件



于是猜测 bin2 也是个 exe 文件，将 bin2 的文件头 64 个字节与 bin1 的文件头 64 个字节异或一下，得到 a0110df809fed085b40d591c6af3cc7ea0110df809fed085b40d591c6af3cc7e


```

28     v14 += 4;
29 }
30 v6[0] = 19295;
31 v6[1] = 57005;
32 v6[2] = 4589;
33 v6[3] = 46028;
34 btea(v7, 11, v6);
35 for ( j = 0; j <= 10; ++j )
36 ;
37 v5[0] = -867866211;
38 v5[1] = 1748849490;
39 v5[2] = -1195675232;
40 v5[3] = 221779885;
41 v5[4] = 1967646622;
42 v5[5] = 1272810620;
43 v5[6] = -1780327807;
44 v5[7] = -1001238269;
45 v5[8] = 1883878511;
46 v5[9] = 1410856828;
47 v5[10] = 1700163896;
48 for ( k = 0; k <= 10; ++k )
49 {
50     if ( v5[k] != v7[k] )
51     {
52         MessageBoxA(0, "error!", &Caption, 0);
53         exit(0);
54     }
55 }
56 MessageBoxA(0, "right!", &Caption, 0);
57 return 0;
58 }

```

跟进 btea

```

10 unsigned int v10; // [esp+1ch] [ebp-Ch]
11 int v17; // [esp+34h] [ebp+Ch]
12
13 if ( a2 <= 1 )
14 {
15     if ( a2 < -1 )
16     {
17         v17 = -a2;
18         v10 = 52 / v17 + 6;
19         v14 = 2033695134 * v10;
20         v16 = *a1;
21         do
22         {
23             v8 = (v14 >> 2) & 3;
24             for ( i = v17 - 2; i; --i )
25             {
26                 v6 = &a1[i];
27                 *v6 -= (((v16 ^ v14) + (a1[i - 1] ^ *(_DWORD *) (4 * (v8 ^ i & 3) + a3))) ^ (((4 * v16) ^ (a1[i - 1] >> 5))
28                     + ((v16 >> 3) ^ (16 * a1[i - 1]))));
29                 v16 = *v6;
30             }
31             *a1 -= (((4 * v16) ^ (a1[v17 - 1] >> 5)) + ((v16 >> 3) ^ (16 * a1[v17 - 1]))) ^ ((v16 ^ v14)
32                 + (a1[v17 - 1] ^ *(_DWORD *) (4 * v8 + a3)));
33             result = *a1;
34             v16 = *a1;
35             v14 -= 2033695134;
36             --v10;
37         } while ( v10 );
38     }
39 }
40 else
41 {
42     00000839 btea:16 (401439)

```

发现是个 xxtea 算法，脚本直接解

```
#include <stdio.h>
```

```
#include <stdint.h>
```

```
#define DELTA 0x7937B99E
```

```
#define MX (((z>>5^y<<2) + (y>>3^z<<4)) ^ ((sum^y) + (key[(p&3)^e] ^ z)))
```

```
void btea(uint32_t *v, int n, uint32_t const key[4])
```

```
{
    uint32_t y, z, sum;
    unsigned p, rounds, e;
    if (n > 1)          /* Coding Part */
    {
        rounds = 52/n;
        sum = 0;
        z = v[n-1];
        do
        {
            sum += DELTA;
            e = (sum >> 2) & 3;
            for (p=0; p<n-1; p++)
            {
                y = v[p+1];
                z = v[p] += MX;
            }
            y = v[0];
            z = v[n-1] += MX;
        }
        while (--rounds);
        printf("\n%x\n",sum);
    }
    else if (n < -1)    /* Decoding Part */
    {
        n = -n;
        rounds = 52/n;
        sum = 0xe4dee678;
        y = v[0];
        do
        {
            e = (sum >> 2) & 3;
            for (p=n-1; p>0; p--)
            {
                z = v[p-1];
                y = v[p] -= MX;
            }
            z = v[n-1];
            y = v[0] -= MX;
            sum -= DELTA;
        }
    }
}
```

```

        while (--rounds);
//        printf("\n%x\n",sum);
    }
}

int main()
{
    uint32_t v[11]= {0xCC45699D, 0x683D5352, 0xB8BB71A0, 0xD3817AD, 0x7547E79E,
0x4BDD8C7C, 0x95E25A81, 0xC4525103, 0x7049B46F, 0x5417F77C, 0x65567138};
    uint32_t const k[4]={0x4B5F,0xDEAD,0x11ED,0xB3CC};
    int i,n= 11; //n 的绝对值表示 v 的长度，取正表示加密，取负表示解密
    // v 为要加密的数据是两个 32 位无符号整数
    // k 为加密解密密钥，为 4 个 32 位无符号整数，即密钥长度为 128 位
    printf("          解          密          前          原          始          数
据: %x %x %x %x %x %x %x %x %x %x\n",v[0],v[1],v[2],v[3],v[4],v[5],v[6],v[7],v[8],v[9],v[10]);
    btea(v, -n, k);
    printf("          解          密          后          的          数
据: %x %x %x %x %x %x %x %x %x %x\n",v[0],v[1],v[2],v[3],v[4],v[5],v[6],v[7],v[8],v[9],v[10]);
    for(i=0;i<11;i++)
    {
        printf("%c",v[i]&0xff);
        printf("%c",v[i]&0xff00)%0xff);
        printf("%c",v[i]&0xff0000)%0xffff);

        printf("%c",v[i]&0xff000000)%0xfffffff);//DASCTF{7eb20cb2-deac-11ed-ae42-94085339ce84
    }
    }
    btea(v, n, k);
    printf("\n          加          密          后          的          数
据: %x %x %x %x %x %x %x %x %x %x\n",v[0],v[1],v[2],v[3],v[4],v[5],v[6],v[7],v[8],v[9],v[10]);
    system("pause");
    return 0;
}

```
