

Разбор заданий CTF

Rabin's problem 100

Очень простая задача.

Дана шифрующая программа на языке Python -

```
OP=open('cipher.txt', 'r')
Cip=open('cipher.enc', 'w')
crypt=""
a=10000000000
for char in OP.read():
    crypt+= str((((ord(char)+a)**2)%1539508836011))+ ' '
    print(crypt)
Cip.write(crypt)
Cip.close()
```

Также дан шифртекст

```
1300995741122 1020995739022 1080995739439 581486913516 101486908188 640995736799
361486910931 201486909203 680995736999 401486911383 21486907412 640995736799
421486911612 21486907412 141486908588 700995737102 421486911612 541486913028
621486914012
```

Пытаемся его расшифровать.

Это можно сделать двумя способами:

1 способ. Факторизуем число 1539508836011 на множители
 $1539508836011 = 1133009 \times 1358779$ самостоятельно реализовав любой метод факторизации, либо воспользовавшись встроенными функциями математических пакетов (например, Wolfram alpha)

Затем реализуется расшифровка для системы Рабина

Из уравнения $y_p p + y_q q = 1$ с помощью расширенного алгоритма Евклида находятся значения y_p и y_q .

Далее используя китайскую теорему об остатках вычисляются следующие 4 значения:

$$m_1 = (y_p p m_q + y_q q m_p) \bmod n$$

$$m_2 = n - m_1$$

$$m_3 = (y_p p m_q - y_q q m_p) \bmod n$$

$$m_4 = n - m_3$$

Сообщение m выбирается из одного из значений m_1, m_2, m_3, m_4 .

2 способ (очень простой). Перебрать значения $b=0, \dots, 256$, чтобы получить значения $c = ((b+a)**2) \% 1539508836011$. Полученные значения c сравнить с символами шифртекста. При совпадении берется b , для которого совпало c и затем с помощью функции $\text{chr}(b)$ получаем шифртекст, т.е. можно установить что:

1300995741122 - R

1020995739022 - D

1080995739439 - G

581486913516 - {

101486908188 - c

640995736799 - 1

361486910931 - p

201486909203 - h

680995736999 - 3

401486911383 - r

21486907412 - _

640995736799 - 1

421486911612 - s

21486907412 - _

141486908588 - e

700995737102 - 4

421486911612 - s

541486913028 - y

621486914012 - }

Ответ: RDG{c1ph3r_1s_e4sy}

A galaxy far, far away....

Дано изображение, которое вроде как выступает в качестве шифра



По названию задачи догадываемся, что это связано со звездными войнами. Гуглим. Узнаем, что существует язык Ауреш со своим собственным алфавитом. Гуглим этот алфавит.

A	B	C	Ch	D	E	Ae	Eo	F	G
aurek	besh	cresh	cherek	dorn	esk	enth	onith	forn	grek
H	I	J	K	Kh	L	M	N	Ng	O
herf	isk	jenth	krill	krenth	leth	mern	nerm	nen	osk
Oo	P	Q	R	S	Sh	T	Th	U	V
orenth	peth	qek	resh	senth	shen	trill	thesh	usk	vev
W	X	Y	Z	,	.	?	!	:	;
wesk	xesh	yirt	zerek						
-	/	'	,	“	”	()	#	
								credits	

Также нужно понять, что заглавные буквы пишутся зеркально приведенному алфавиту. Декодируем сообщение и получаем:

UGJ{Pdb_wkh_4wk_eh_zlwk_brx}

В результате анализа на соответствие простейшим шифрам получаем, что это шифр Цезаря (сдвиг 3 символа).

Расшифровываем.

Ответ: RDG{May_the_4th_be_with_you}

Secret square

У нас дан текстовый файл со следующим сообщением:

[illegible]

Понимаем, что символы образуют квадрат 25x25. Понимаем, что W - обозначает белый цвет, а B - черный.

Восстанавливаем картинку



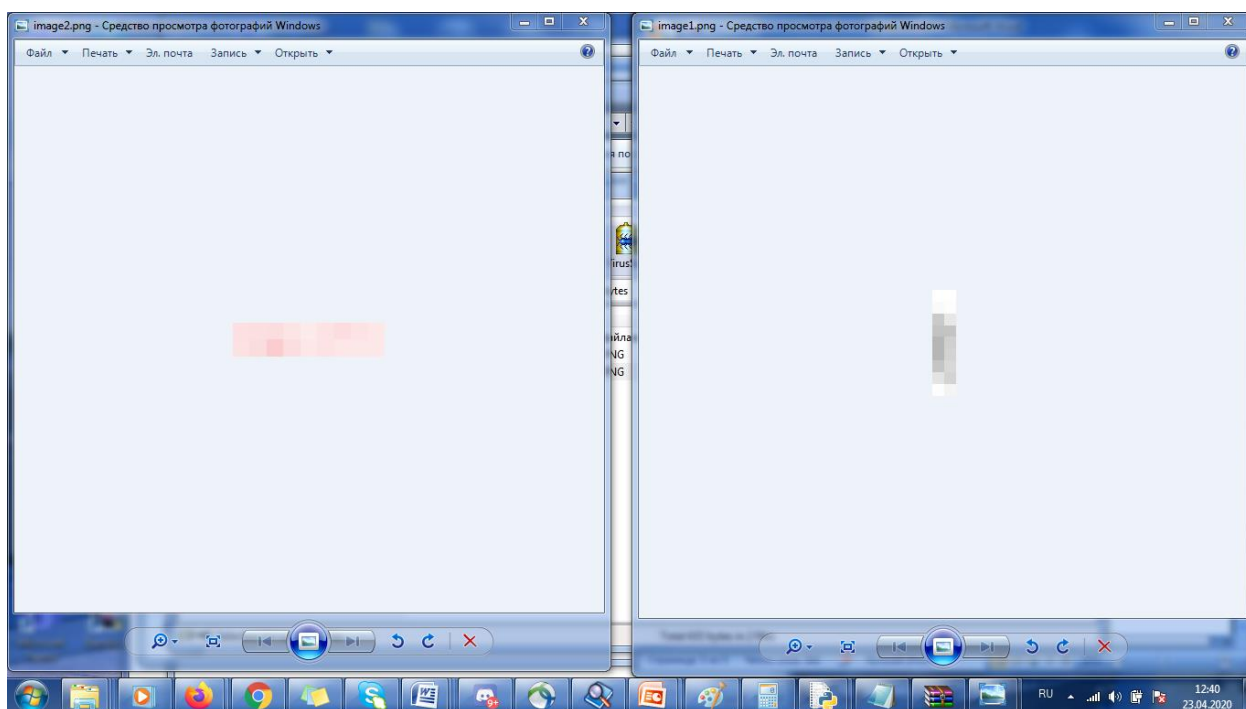
Декодируем QR-код.

Ответ: RDG{0ffic3_d0cum3nt}

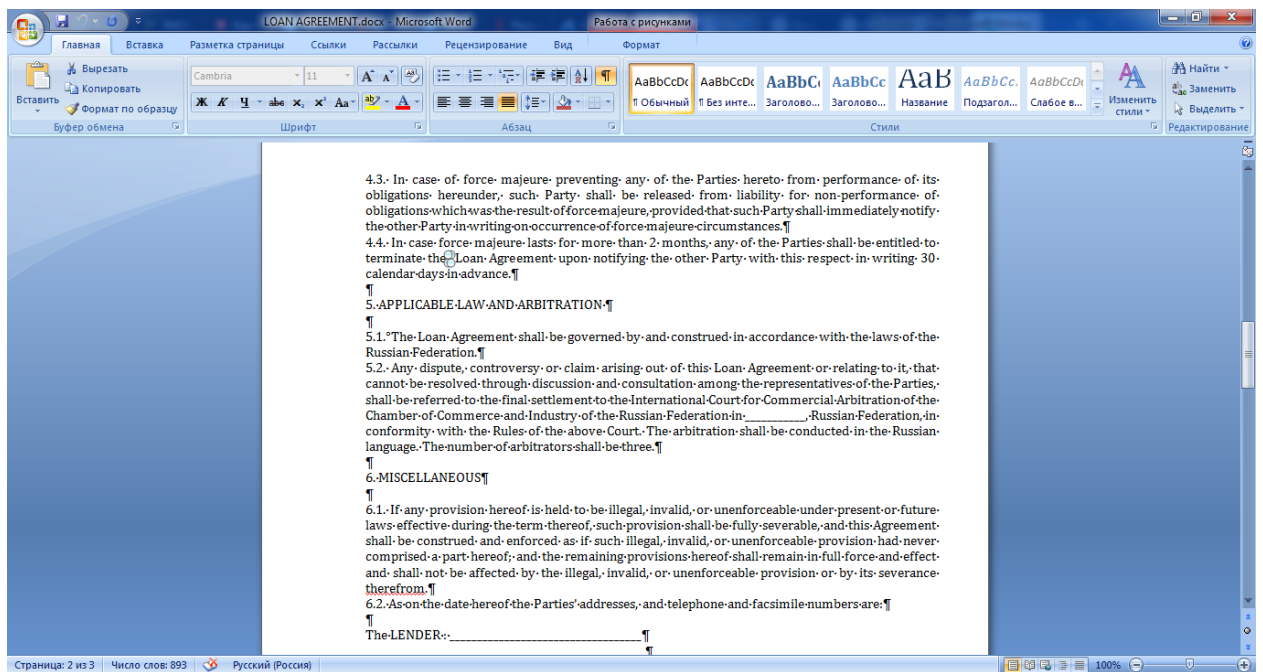
Contract

Дан тестовый документ. Открываем его с помощью winrar.

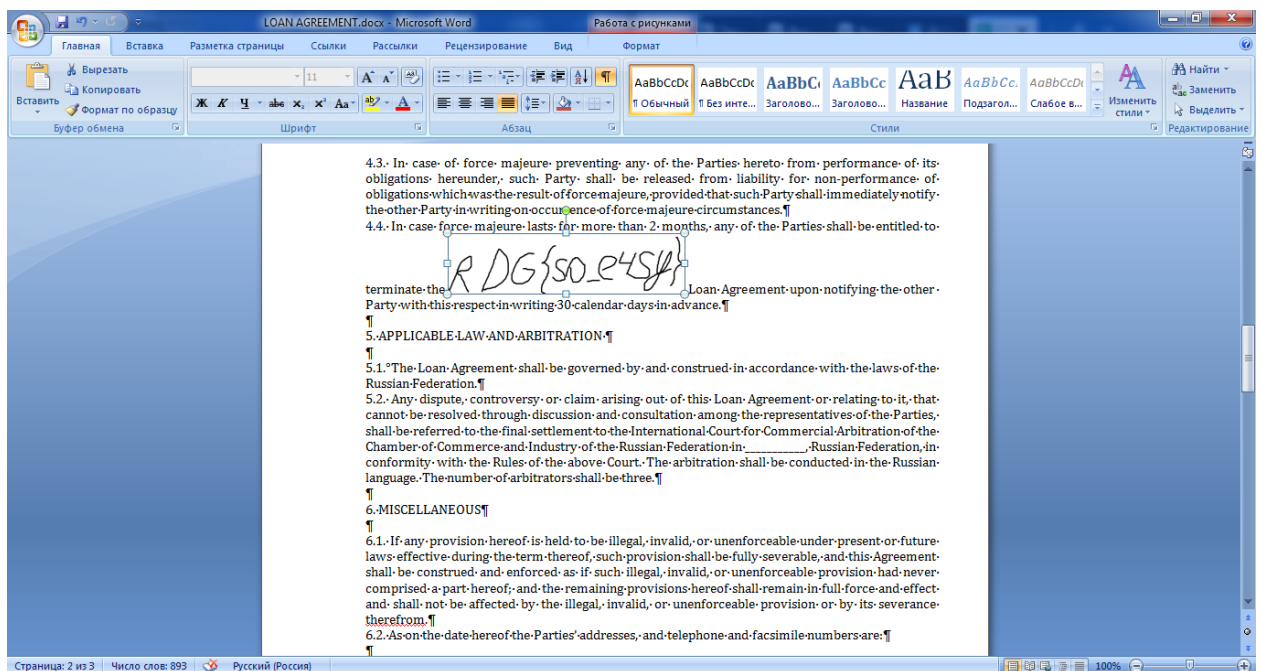
Видим 2 размазанных изображения



Открываем сам документ. И находим место где сокрыто изображение (оно одно, так как первое было ранее удалено).



Разворачиваем в текстовом файле найденное изображение и находим флаг

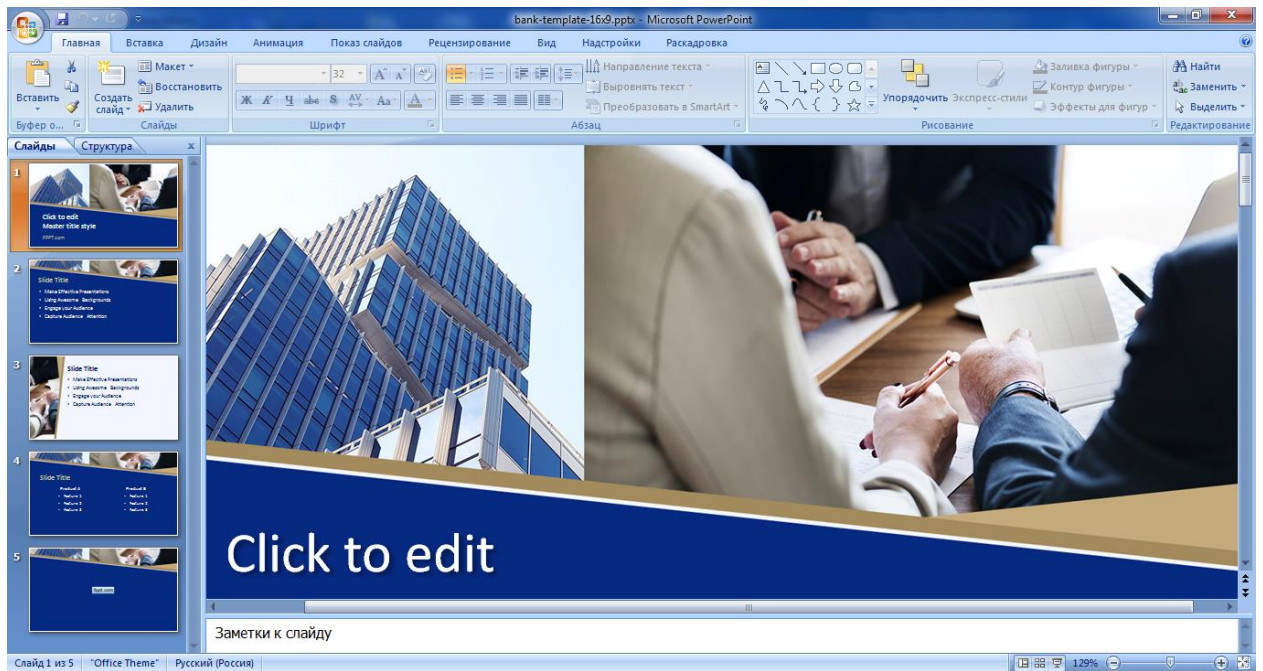


Ответ: RDG{so_e4sy}

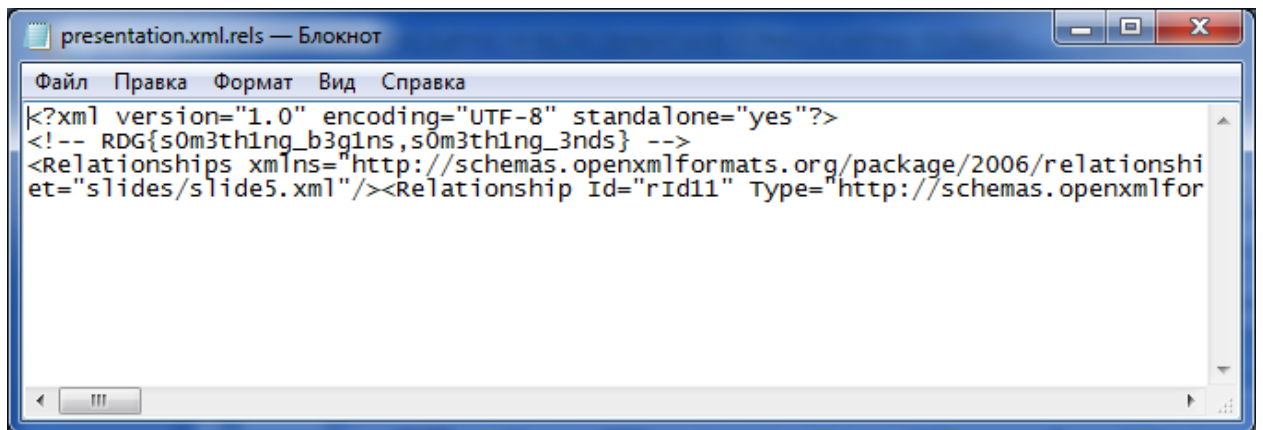
Presentation

Дана презентация.

При ее открытии мы ничего не находим.



Откроем ее с помощью winrar. Просматриваем ее элементы. Доходим до элемента presentation.xml.rels . Открываем его и находим флаг.



Ответ: RDG{s0m3th1ng_b3g1ns,s0m3th1ng_3nds}

HLF me, baby

Abstract: When real hackers get down to business, extremely sophisticated tools can come into play. When the villains decided to attack our system, we managed to intercept a special decoder designed to handle some kind of logs. Unfortunately, the log itself was damaged, and no one can crack the decoder. Now only you can help us deal with this problem.

При "холостом" запуске ничего понять нельзя.

```
C:\Workdir\Writeps_RDG\HLF>hacker_log_reader.exe
C:\Workdir\Writeps_RDG\HLF>
```

Кормим файл IDA, видим следующий код в main:

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    unsigned int v3; // eax
    const wchar_t *v4; // esi
    DWORD v5; // eax
    FILE *v6; // eax
    FILE *v7; // edi
    int v8; // ebx
    _OWORD *v9; // eax
    _OWORD *v10; // esi
    DWORD *v11; // eax
    int v12; // ecx

    v3 = sub_4054CF(0);
    srand(v3);
    if ( argc != 2 )
        goto LABEL_16;
    v4 = (const wchar_t *)argv[1];
    v5 = GetFileAttributesW((LPCWSTR)argv[1]);
    if ( v5 == -1 )
        goto LABEL_16;
    if ( v5 & 0x10 )
        goto LABEL_16;
    v6 = _wopen(v4, L"rb");
    v7 = v6;
    if ( !v6 )
        goto LABEL_16;
    fseek(v6, 0, 2);
    v8 = ftell(v7);
    fseek(v7, 0, 0);
    v9 = malloc(v8 + 1);
    v10 = v9;
    if ( !v9 || (memset(v9, 0, v8), sub_4056F5(v10, 1, v8, (int)v7), fclose(v7), (v11 = malloc(0x16u)) == 0) )
        goto LABEL_16;
    ExitProcess(0xFFFFFFFF);
    v12 = *((_DWORD *)v10 + 4);
    *(_OWORD *)v11 = *v10;
    v11[4] = v12;
    *(_BYTE *)v11 + 20 = *(_BYTE *)v10 + 20;
    if ( _byteswap_ulong(*v11) == 1212958208
        && *(_BYTE *)v11 + 4 >= 2u
        && _byteswap_ulong(*(_DWORD *)((char *)v11 + 5)) == v8 )
    {
        *(_DWORD *)((char *)v10 + 9) = 0;
        if ( _byteswap_ulong(*(_DWORD *)((char *)v11 + 9)) == sub_401150(v10, v8) )
            sub_401180((char *)v10 + 21, v8 - 21);
    }
    return 0;
}
```

Приводим декомпилированный листинг к читаемому виду:


```

const wchar_t *logFilePath; // esi
DWORD nFileAttr; // eax
FILE *fptr; // eax
FILE *_fptr; // edi
DWORD nFileSize; // ebx
__OWORD *pFileBuffer; // eax
__OWORD *_pFileBuffer; // esi
__DWORD *_pfBuffer; // eax
int v12; // ecx

nTimeParam = Time(0);
srand(nTimeParam);
if ( argc != 2 )
    goto Error_label;
logFilePath = (const wchar_t *)argv[1];
nFileAttr = GetFileAttributesW((LPCWSTR)argv[1]); // check file existence
if ( nFileAttr == -1 )
    goto Error_label;
if ( nFileAttr & 0x10 )
    goto Error_label;
fptr = _wfopen(logFilePath, L"rb"); // open file (if exists)
_fptr = fptr;
if ( !fptr )
    goto Error_label;
fseek(fptr, 0, 2);
nFileSize = ftell(_fptr); // Get file size
fseek(_fptr, 0, 0);
pFileBuffer = malloc(nFileSize + 1); // allocate memory for file contents
_pFileBuffer = pFileBuffer;
if ( !pFileBuffer
    || (memset(pFileBuffer, 0, nFileSize),
        sub_4056F5(_pFileBuffer, 1, nFileSize, (int)_fptr),
        fclose(_fptr),
        (__pfBuffer = malloc(0x16u)) == 0) )
{
Error_label:
    ExitProcess(0xFFFFFFFF);
}
v12 = *((__DWORD *)_pFileBuffer + 4); // IDA's decompiler is not perfect as you see
*(__OWORD *)__pfBuffer = *_pFileBuffer;
__pfBuffer[4] = v12;
*((__BYTE *)__pfBuffer + 20) = *((__BYTE *)_pFileBuffer + 20);
if ( _byteswap_ulong(*__pfBuffer) == 0x484C4600 // check file params
    && *((__BYTE *)__pfBuffer + 4) >= 2u
    && _byteswap_ulong((__DWORD *)((char *)__pfBuffer + 5)) == nFileSize ) // check file size
{
    *(__DWORD *)((char *)_pFileBuffer + 9) = 0;
    if ( _byteswap_ulong((__DWORD *)((char *)__pfBuffer + 9)) == sub_401150(_pFileBuffer, nFileSize) )
        sub_401180((char *)_pFileBuffer + 21, nFileSize - 21);
}
return 0;
}

```

Из листинга выше можно понять, что в main сначала проверяется наличие файла-лога, затем содержимое файла читается в саллоцированный буффер и идет некая проверка значений:

- первые 4 байта сравниваются с константным значением 0x484c4600
- пятый байт сравнивается с двойкой
- DWORD по смещению 5 сравнивается с реальным размером файла

Если проверки пройдены, зануляются 4 байта по оффсету 0x9 и содержимое вместе с размером передается в функцию sub_401150:

```

int __cdecl sub_401150(char *a1, int a2)
{
    int v2; // edx
    unsigned int i; // eax
    char v5; // cl

    v2 = a2;
    for ( i = -1; v2; --v2 )
    {
        v5 = *a1++;
        i = dword_41A998[(unsigned __int8)(i ^ v5)] ^ (i >> 8);
    }
    return ~i;
}

```

Смотрим, что лежит в массиве dword_41A998:

```

dword_41A998 dd 0, 77073096h, 0EE0E612Ch, 9909518Ah, 76DC419h, 706AF48Fh
; DATA XREF: sub_401150+1F1r
dd 0E963A535h, 9E6495A3h, 0ED88832h, 79DCB8A4h, 0E0D5E91Eh
dd 97D2D988h, 9B64C2Bh, 7EB17C8Dh, 0E7882D07h, 90BF1D91h
dd 1DB71064h, 6AB020F2h, 0F3B97148h, 84BE41DEh, 1ADAD47Dh
dd 6DDDE4EBh, 0F4D48551h, 83D385C7h, 136C9856h, 646BA8C0h
dd 0FD62F97Ah, 8A65C9ECh, 14015C4Fh, 63066CD9h, 0FA0F3D63h
dd 8D080DF5h, 3B6E20C8h, 4C69105Eh, 0D56041E4h, 0A2677172h
dd 3C03E4D1h, 4B04D447h, 0D20D85FDh, 0A50A856Bh, 35B5A8FAh
dd 42B2986Ch, 0DB8BC9D6h, 0ACBCF940h, 32D86CE3h, 45DF5C75h
dd 0DCD60DCFh, 0ABD13D59h, 26D930ACH, 51DE003Ah, 0C8D75180h
dd 08FD06116h, 21B4F485h, 56B3C423h, 0CFBA9599h, 088BDA50Fh
dd 2802B89Eh, 5F058808h, 0C60CD9B2h, 0B10BE924h, 2F6F7C87h
dd 58684C11h, 0C1611DABh, 0B6662D3Dh, 76DC4190h, 1DB7106h
dd 98D220BCh, 0EFD5102Ah, 71B18589h, 6B6B51Fh, 9FBFE4A5h
dd 0E888D433h, 7807C9A2h, 0F00F934h, 9609A88Eh, 0E10E9818h
dd 7F6A0DB8h, 86D3D2Dh, 91646C97h, 0E6635C01h, 6B6B51F4h
dd 1C6C6162h, 856530D8h, 0F262004Eh, 6C0695EDh, 1B01A57Bh
dd 8208F4C1h, 0F50FC457h, 65B0D9C6h, 12B7E950h, 8BBE8E8Ah
dd 0FCB9887Ch, 62DD1DDFh, 15DA2D49h, 8CD37CF3h, 0FBD44C65h
dd 4DB26158h, 3AB551CEh, 0A3BC0074h, 0D48B30E2h, 4ADFA541h
dd 3DD895D7h, 0A4D1C46Dh, 0D3D6F4FBh, 4369E96Ah, 346ED9FCh
dd 0AD678846h, 0DA60B8D0h, 44042D73h, 33031DE5h, 0AA0A4C5Fh
dd 0DD0D7CC9h, 5005713Ch, 270241AAh, 0BE0B1010h, 0C90C2086h
dd 57688525h, 206F8583h, 0B966D409h, 0CE61E49Fh, 5EDEF90Eh
dd 29D9C998h, 0B0D09822h, 0C7D7A884h, 59B33D17h, 2EB40D81h
dd 0B7BD5C3Bh, 0C0BA6CADh, 0ED888320h, 9ABFB3B6h, 3B6E20Ch
dd 7481D29Ah, 0EAD54739h, 9DD277AFh, 4DB2615h, 73DC1683h
dd 0E3630B12h, 94643884h, 0D6D6A3Eh, 7A6A5AA8h, 0E40ECF0Bh
dd 9309FF9Dh, 0A00AE27h, 7D079EB1h, 0F00F9344h, 8708A3D2h
dd 1E01F268h, 6906C2FEh, 0F762575Dh, 806567CBh, 196C3671h
dd 6E6B06E7h, 0FED41B76h, 89D32BE0h, 10DA7A5Ah, 67DD4ACCh
dd 0F9B9DF6Fh, 8EBEEFF9h, 17B7BE43h, 60B08ED5h, 0D6D6A3E8h
dd 0A1D1937Eh, 38D8C2C4h, 4FDF252h, 0D1B867F1h, 0A6BC5767h
dd 3FB506DDh, 48B23648h, 0D80D2BDAh, 0AF0A1B4Ch, 36034AF6h
dd 41047A60h, 0DF60EFC3h, 0A867DF55h, 316E8EEFh, 4669BE79h
dd 0CB61B38Ch, 0BC66831Ah, 256FD2A0h, 5268E236h, 0CC0C7795h
dd 0BB0B4703h, 220216B9h, 5505262Fh, 0C5BA3BBEh, 0B2BD0B28h
dd 2BB45A92h, 5CB36A04h, 0C2D7FFA7h, 0B5D0CF31h, 2CD99E8Bh
dd 5BDEAE1Dh, 9B64C2B0h, 0EC63F226h, 756AA39Ch, 26D930Ah
dd 9C0906A9h, 0EB0E363Fh, 72076785h, 5005713h, 95BF4A82h
dd 0E2B87A14h, 7BB12BAEh, 0CB61B38h, 92D28E98h, 0E5D58E0Dh
dd 7CDCEFB7h, 0BD8DF21h, 86D3D2D4h, 0F1D4E242h, 68DD83F8h
dd 1FDA836Eh, 81BE16CDh, 0F6B9265Bh, 6FB077E1h, 18B74777h
dd 88085AE6h, 0FF0F6A70h, 66063BCAh, 11010B5Ch, 8F659EFFh
dd 0F862AE69h, 616BFFD3h, 166CCF45h, 0A00AE278h, 0D70DD2EEh
dd 4E048354h, 3903B3C2h, 0A7672661h, 0D06016F7h, 4969474Dh

```

Немного погуглив, видим, что константы относятся к подсчету CRC32 хэша.

В конце main вызывается функция sub_401180, которая принимает два аргумента:

- 1) Указатель на буффер без заголовка (0x15 байт)
- 2) Размер содержимого без заголовка

Проанализировав функцию sub_401180 кроме весьма невятного алгоритма дешифрования видим следующие строки:

```
sub_401010("%s\n", v10);  
ExitProcess(0);
```

Таким образом, алгоритм работы можно описать как-то так:

1. Проверяется наличие лог-файла
2. Проверяется заголовок лог-файла (константное значение-magic, длина файла, CRC)
3. при корректном заголовке содержимое дешифруется и выводится в консоль

****** Примечательно также, что байты по смещения [0x9 - 0x15] не проверяются.

Далее есть три варианта вероятных действий:

- Добавить заголовок (недостающие байты)
- Изменить текущий corrupted лог
- Попытаться восстановить алгоритм шифрования

Наиболее простым выглядит вариант #1 (на самом деле вариант 2 неверен, а 3 достаточно трудоемкий что так или иначе приведет нас именно к такому пути решения), так что просто добавим недостающие байты в любом hex редакторе, а потом "скормим" файл программе:

****** *Недостающий 8-байтовый промежуток можно заполнить любыми значениями, проверки в коде на корректность этих данных нет.*

```
C:\Workdir\Writes_RDG\HLF>hacker_log_reader.exe corrupted  
delightful remarkably mr on announcing themselves entreaties favourable. About to in so terms voice at.  
friendship one sufficient terminated frequently themselves. It more shed went up is roof if loud case.  
ly enough passed is up. You disposal strongly quitting his endeavor two settling him. Manners ham him h  
our more part. Adapted as smiling of females oh me journey exposed concern. Met come add cold calm rose  
anny. Discretion at be an so decisively especially. Exeter itself object matter if on mr in. So insisted  
eady to which up. Attacks smiling and may out assured moments man nothing outward. Thrown any behind aff  
melancholy in acceptance collecting frequently be if. Zealously now pronounce existence add you instant  
Concerns no in expenses raillery formerly. Bringing unlocked me an striking ye perceive. Mr by wound ho  
ing we. Most my no spot felt by no. He he in forfeited furniture sweetness he arranging. Me tedious so  
objection for elsewhere her preferred allowance her. Marianne shutters mr steepest to me. Up mr ignorant  
Ham whom call all lain like. Among going manor who did. Do ye is celebrated it sympathize considered.  
age. Own her miss cold last. It so numerous if he outlived disposal. How but sons mrs lady when. Her esp  
unreserved resolution. Hence hopes noisy may china fully and. Am it regard stairs branch thirty length  
. Entrance prospect removing we packages strictly is no smallest he. For hopes may chief get hours day  
t. Forbade few through inquiry blushes you. Cousin no itself eldest it in dinner latter missed no. Bois  
ction friendship say boy. Him mrs shy article smiling respect opinion excited. Welcomed humoured rejoic  
satisfiable so if he entreaties appearance. Rose you wife how set lady half wish. Hard sing an in true fel  
table finished of oh. Entered at excited at forming between so produce. Chicken unknown besides attacks  
favourable on reasonably melancholy estimating. Own hence views two ask right whole ten seems. What nea  
fficient why pianoforte. Unwilling sportsmen he in questions september RDG{H4Te L0Gs Br00} therefore des  
as. Reasonably how possession shy way introduced age inquietude. Missed he engage no exeter of. Still tr  
can design tastes did joy settle. Roused future he ye an marked. Arose mr rapid in so vexed words. Gay  
andered relation no surprise of screened doubtful. Overcame no insisted ye of trifling husbands. Might
```

ReverseMe

При запуске файла требуется какой-то user-input (пока неизвестно, как он влияет на ход исполнения и влияет ли вообще). Посмотрим что выдаст декомпилятор:

```

signed int v7; // kr00_4d/
int v8; // esi@9
int v9; // eax@9
int v10; // edi@9
int v11; // eax@10
int v12; // edi@10
int v13; // edi@11
unsigned int v14; // edx@12
int v15; // ecx@13
signed int v16; // kr04_4@16
signed int v17; // ebx@16
signed int v18; // ecx@16
int v19; // edx@17
int v21; // [sp+0h] [bp-320h]@9
int v22; // [sp+4h] [bp-31Ch]@10
int v23; // [sp+8h] [bp-318h]@3
int v24; // [sp+Ch] [bp-314h]@5
unsigned int v25; // [sp+10h] [bp-310h]@1
WCHAR Filename; // [sp+14h] [bp-30Ch]@1
char v27[256]; // [sp+21Ch] [bp-104h]@1

sub_402190(v27, 0, 256);
sub_402190(&Filename, 0, 520);
v25 = 0;
if ( GetModuleFileNameW(0, &Filename, 0x104u) ==
ABEL_2:
    ExitProcess(0xFFFFFFFF);
v0 = PathFindFileNameA((LPCSTR)&Filename);
v1 = v0;
*PathFindExtensionA(v0) = 0;
sub_401050("%255s", (unsigned int)v27);
v2 = sub_404E77(v1, L"rb");
v23 = _DWORD(const char *, ...)
if ( v2 )
{

```

Функция в IDA 6.8 выглядит достаточно неприятно. Переименуем известные функции (многие из них читаются по аргументам) и разобьем на две части ввиду громоздкости алгоритма:

```

memset(Src, 0, sizeof(Src));
memset(Filename, 0, sizeof(Filename));
Size = 0;
if ( GetModuleFileNameW(0, Filename, 0x104u) == 260 )// check file existence
ABEL_2:
    ExitProcess(0xFFFFFFFF);
v3 = PathFindFileNameA((LPCSTR)Filename);
v4 = (const wchar_t *)v3;
*PathFindExtensionA(v3) = 0;
getchh("%255s", Src); // some unused getch
fptr = _wfpopen(v4, L"rb"); // openc current binary for reading
Stream = fptr;
if ( fptr )
{
    fseek(fptr, -4, 2); // set pile pointer to END-4
    if ( fread(&Size, 1, 4, (int)Stream) > 0 ) // read last 4 bytes
    {
        Size = (((Size << 16) | Size & 0xFF00) << 8) | (((Size >> 16) | Size & 0xFF0000) >> 8); // reverse previously checked bytes
        allocBuffer = malloc(Size); // allocate mem of prev read size
        _allocBuffer = allocBuffer;
        _allocBuffer = allocBuffer;
        if ( allocBuffer )
        {
            _Size = Size;
            memset(allocBuffer, 0, Size); // zero buffer
            fseek(Stream, -4 - _Size, 2); // Set File pointer to SIZE (read) - 4
            v9 = fread(_allocBuffer, 1, Size, (int)Stream); // read again
            if ( v9 == Size )
            {
                v10 = strlen(Src);

```

В первой части программа получает путь к самой себе и открывает себя в режиме чтения. После этого читаются первые 4 байта “хвоста”, а затем байты переворачиваются и полученное значение служит размером для нового буфера. Далее с конца (за исключением первых 4х байт) читается чанк данных и сохраняется в ранее саллоцированный буфер.

```
if ( v9 == Size )
{
    inputlen = strlen(input_val);
    if ( inputlen & 1 )
        ExitProcess(0xFFFFFFFF);
    HalfInputlen = inputlen / 2;
    halfBuffer = malloc(inputlen / 2 + 1);
    v13 = halfBuffer;
    v24 = halfBuffer;
    if ( halfBuffer )
    {
        memset(halfBuffer, 0, inputlen / 2);
        memmove(v13, input_val, inputlen / 2);
        v14 = malloc(HalfInputlen + 1);
        v15 = v14;
        v25 = v14;
        if ( v14 )
        {
            memset(v14, 0, inputlen / 2);
            memmove(v15, &input_val[HalfInputlen], inputlen / 2);
            v16 = (char *)malloc(inputlen);
            memmove(v16, v25, inputlen / 2);
            memmove(&v16[HalfInputlen], v24, inputlen / 2);
            if ( inputlen != 18 )
                goto LABEL_2;
            for ( i = 0; i < 0x12; ++i )
            {
                v18 = (unsigned __int8)v16[i];
                constant_ba[i] += i & 1 | 0xA;
                if ( constant_ba[i] != v18 )
                    goto LABEL_2;
            }
            v16[i] = 0;
            if ( v16 )
            {
                v19 = strlen(input_val);
                v20 = Size;
                sub_401020((int)"%d\n", Size);
                v21 = 0;
                if ( v20 <= 0 )
                {
                    v22 = __allocBuffer;
                }
                else
                {
                    do
                    {
                        v22 = __allocBuffer;
                        __allocBuffer[v21] ^= v16[v21 % v19]; // xor algo
                        ++v21;
                    } while ( v21 < v20 );
                }
            }
        }
    }
}
```

Во второй части алгоритма ранее полученное через user-input строковое значение делится на 2 и по частям сравнивается с константным массивом constant_ba. Кроме того значения массива в процессе проверки складываются с 0xA или 0xB в зависимости позиции буквы (четная / нечетная):

```
; char constant_ba[]
constant_ba    db 5Eh, 29h, 59h, 60h, 55h, 62h, 58h, 54h, 6Ah, 25h, 55h
                                     ; DATA XREF: sub_401080+23A1↑w
                                     ; sub_401080+2401↑r
                db 69h, 5Eh, 5Ah, 55h, 28h, 64h, 59h
                dd      0
```

В общем виде функцию проверки пользовательского ключа можно представить следующим кодом:

```
char real_pwd[] = {
0x5e, 0x29, 0x59, 0x60, 0x55, 0x62, 0x5b, 0x54, 0x6a, 0x25, 0x55, 0x69, 0x5e, 0x5a, 0x55, 0x
28, 0x64, 0x59 };

BYTE* CheckPassword(char* input_pwd, int size) {
    BOOL nResult = FALSE;
    BYTE* pwd_part_1 = NULL;
    BYTE* pwd_part_2 = NULL;
    BYTE* final_pwd = NULL;
```

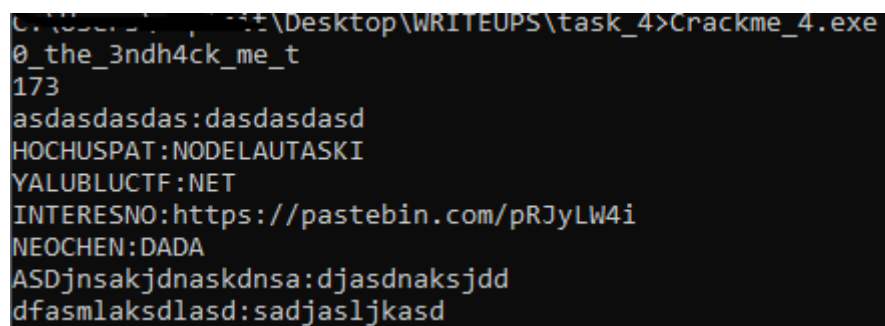
```

// 1) check if size is dividable by 2
if (size % 2 != 0) {
ExitProcess(-1);
}
// 2) divide pwd
pwd_part_1 = malloc(size / 2 + 1);
if (pwd_part_1) {
memset(pwd_part_1, 0, size / 2);
memcpy(pwd_part_1, input_pwd, size / 2);
pwd_part_2 = malloc(size / 2 + 1);
if (pwd_part_2) {
memset(pwd_part_2, 0, size / 2);
memcpy(pwd_part_2, input_pwd + size / 2, size / 2);
final_pwd = malloc(size);
memcpy(final_pwd, pwd_part_2, size / 2);
memcpy(final_pwd + (size / 2), pwd_part_1, size / 2);
if (size != sizeof(real_pwd)) {
ExitProcess(-1);
}
int i = 0;
for (i = 0; i < sizeof(real_pwd); i++) {
if (i % 2 == 0) {
real_pwd[i] = real_pwd[i] + 0xA;
}
else {
real_pwd[i] = real_pwd[i] + 0xB;
}
if (real_pwd[i] != final_pwd[i]) {
ExitProcess(-1);
}
}
final_pwd[i] = '\0';
}
return final_pwd;
}

```

Обратным преобразованием получаем ключ (достаточно просто выставить точки останова на последней итерации цикла-проверки): 0_the_3ndh4ck_me_t

Подаем ключ в программу:



```

C:\Users\...\Desktop\WRITEUPS\task_4>Crackme_4.exe
0_the_3ndh4ck_me_t
173
asdasdasdasd:dasdasdasd
HOCHUSPAT:NOBELAUTASKI
YALUBLUCTF:NET
INTERESNO:https://pastebin.com/pRJyLW4i
NEOCHEN:DADA
ASDjnsakjdnaskdlsa:djasdnaksjdd
dfasmlaksdlasd:sadjasljkasd

```

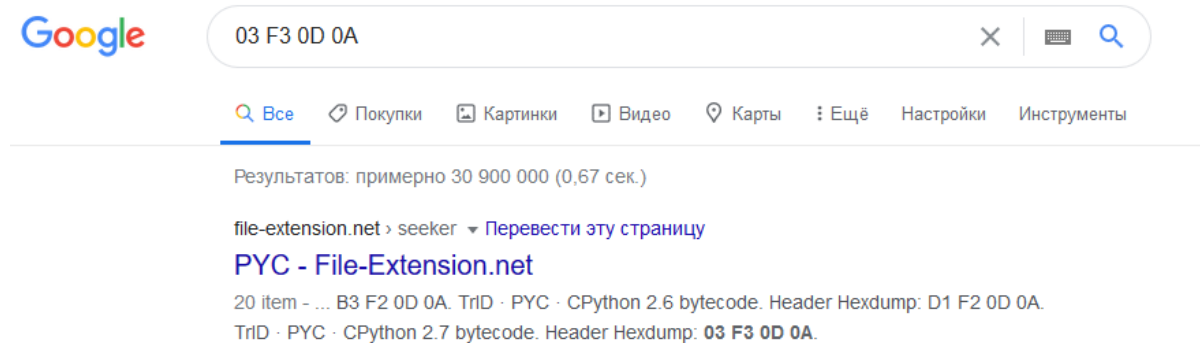
По ссылке на pastebin находим флаг: RDG{w1sh_y0_luck_m333n}

** на данный момент ссылка уже сдохла, но если вы дошли до этого этапа, думаю, дальше сложностей не возникнет

A smile of a snake

Abstract: Our brave hackers got access to the computer of the secretary of one of the local officials. In addition to the fact that the girl uses outdated technologies, she also keeps her secrets in some interesting format ...

Перед нами файл неизвестного формата, берем первые 4 байта и пишаем в гугл:



Пишаем файл в любой python-декомпилятор, который понимает это расширение:

```
task_3.pyc_dis X
1 # Embedded file name: task.py
2 import random
3 sadasd = 111
4 fkycpp = 110
5 mnlouy = 103
6 psnhut = 108
7
8 def jgodmvy(abcdsdad, abcdsdaad):
9     abcdsdad = len(abcdsdaad)
10    return bytearray((abcdsdad[abcdsdad] ^ abcdsdaad[abcdsdad % abcdsdad])
11
12
13 aaaaaaaa = ''
14 aa = [8,
15     6,
16     67,
17     14,
18     0,
19     13,
20     21,
21     2]
22 abcdsdad = [25,
23     22,
24     26,
25     65,
26     45,
27     31,
28     18,
29     22,
30     24,
31     86,
32     31,
33     55,
34     21,
35     82,
36     52,
37     22,
38     26,
39     28,
40     2,
41     84,
42     0,
43     29,
44     13]
45 asdlas = psnhut
46 aaaaaaaa = chr(asdlas)
```

Перед нами слегка обфусцированный python-скрипт. Используя любой удобный текстовый редактор меняем названия переменных и получаем вот такой код:

```
aaaaaaaa = ''
aa = [8, 6, 67, 14, 0, 13, 21, 2]
abcdssdad = [25, 22, 26, 65, 45, 31, 18, 22, 24, 86, 31, 55, 21, 82, 52, 22, 26]
aaaaaaaa = chr(108)
aaaaaaaa += chr(111) + chr(110) + chr(103)
flag = [39, 54, 34]
aaaaaaaa += chr(108) + chr(111) + chr(110) + chr(103)
aaaaaaaa += chr(115)
for mvcbncvb in aa:
    flag.append(mvcbncvb)

aaaaaaaa += chr(110) + chr(97) + chr(107) + chr(101)
aaaaaaaa = []
key = [117, 114, 101, 115, 117, 112, 101, 114, 104, 97, 99, 107, 101, 114] # u

for mvcbncvb in abcdssdad:
    flag.append(mvcbncvb)

print (flag) ## !!!!! ADD THIS FUNCTION !!!!!

aaaa = aaa * 2
aaaa = aaaa + 10
aaaa = aaaa / 2
aaaa = aaaa - aaa
print ' ' + str(aaa)
aaaaaaaaaaaa = raw_input(' ')
if aaaaaaaaaaaaa == aaaaaaaaa:
    pass
else:
    exit()
aaaaaaaaaaaaaa = raw_input(' ') #input key here
for aaaaa in aaaaaaaaaaaaa:
    for aaaaaaaaaaaaa in key:
        if aaaaa == str(chr(aaaaaaaaaaaaa)):
            print ' '
            aaaaaaaaaaaaaaa = xor_func(flag, key)
            print aaaaaaaaaaaaaaa
            break
        else:
            print ' '
            exit(0)
```

Алгоритм тривиален, нет никакой необходимости полностью реконструировать скрипт. Флаг предварительно похороненный константным ключом уже лежит в скрипте, программа принимает от пользователя ключ и пытается сделать обратное преобразование.

В данном случае нам необходимо лишь знать полное содержание массива с флагом, а затем вызвать функцию-декриптор . Таск решается приведением к подобному виду:

```
# Embedded file name: task.py
import random

def xor_func(abcdssdad, abcdsdaad):
    abcdssdad = len(abcdsdaad)
    return bytearray((abcdssdad[abcdssdad] ^ abcdsdaad[abcdssdad % abcdssdad] for abcdssdad in range(0, len(abcdssdad))))

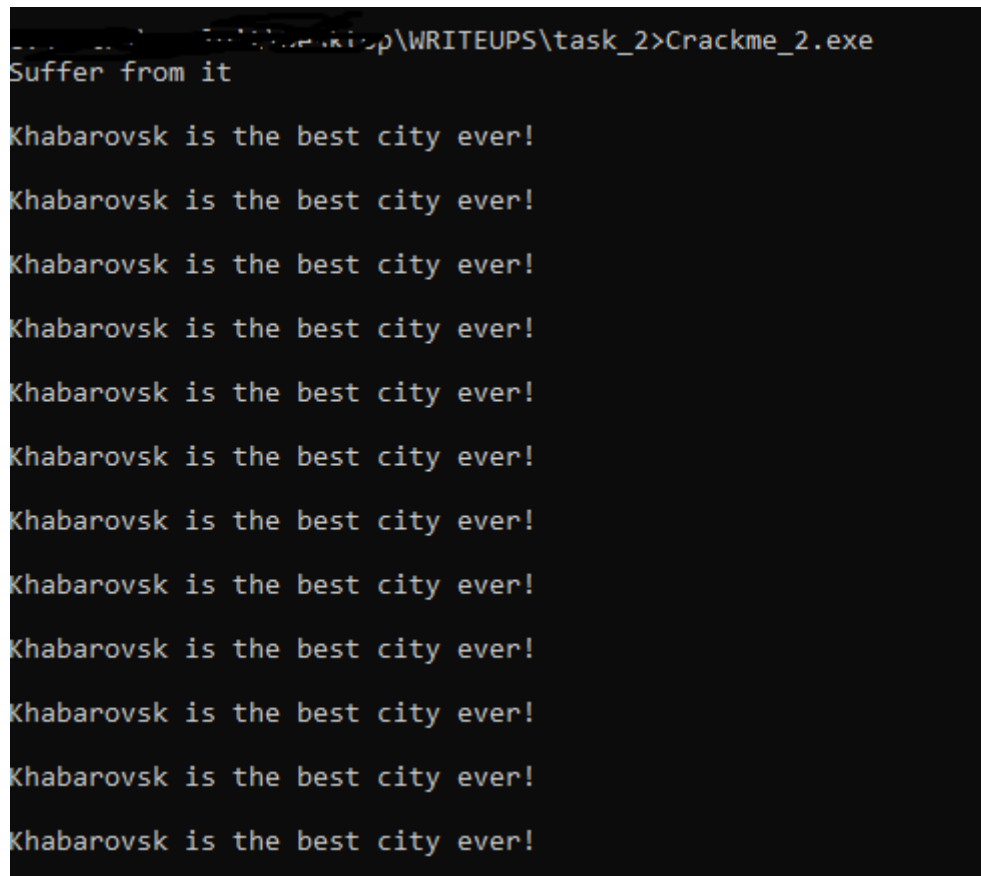
key = [117, 114, 101, 115, 117, 112, 101, 114, 104, 97, 99, 107, 101, 114] # urssuperhacker
flag = [39, 54, 34, 8, 6, 67, 14, 0, 13, 21, 2, 25, 22, 26, 65, 45, 31, 18, 22, 24, 86, 31, 55, 21, 82, 52, 22, 26, 28, 2, 84, 0, 29, 13]
flag_rl = xor_func(flag, key)
print (flag_rl)

RDG{s3kretarsh4_zach3m_t1_ship1sh}
```


Streetstyler

Abstract: There are many street artists in our city! By the way, some of them are fond of hacking.

Перед нами exe файл. При запуске – нагло врёт:



```
... \Desktop\WRITEUPS\task_2>Crackme_2.exe
Suffer from it

Khabarovsk is the best city ever!
Khabarovsk is the best city ever!
Khabarovsk is the best city ever!
Khabarovsk is the best city ever!
Khabarovsk is the best city ever!
Khabarovsk is the best city ever!
Khabarovsk is the best city ever!
Khabarovsk is the best city ever!
Khabarovsk is the best city ever!
Khabarovsk is the best city ever!
Khabarovsk is the best city ever!
Khabarovsk is the best city ever!
```

Открываем бинарник в IDA:

По строкам ищем точку, через которую проходит исполнение:

```
signed int sub_401040()
{
    signed int result; // eax@2
    signed int v1; // esi@3

    if ( IsDebuggerPresent() )
    {
        sub_4010B0(&unk_41A598, 124);
        result = 1;
    }
    else
    {
        sub_401010((const char *)&unk_41770C, (unsigned int)"Suffer from it\n");
        Sleep(0x64u);
        v1 = 100;
        do
        {
            sub_401010((const char *)&unk_41770C, "Khabarovsk is the best city ever!\n");
            Sleep(0x64u);
            --v1;
        }
        while ( v1 );
        result = 0;
    }
    return result;
}
```

Протренировав данную функцию убеждаемся, что для триггера нового пути исполнения необходимо пропатчить проверку `IsDebuggerPresent()`:

```
Khabarovsk_function proc near                                ; CODE XREF: sub_C810B0+1
    call     ds:IsDebuggerPresent
    test     eax, eax
    jz       short loc_C8105F
    mov      edx, 7Ch
    mov      ecx, offset unk_C9A598
    call     sub_C810B0
    mov      eax, 1
    retn

; -----

loc_C8105F:                                                  ; CODE XREF: Khabarovsk_function+1
    push     esi
    push     edi
    push     offset aSufferFromIt ; "Suffer from it"
    push     offset unk_C9770C
    call     sub_C81010
    mov      edi, ds:Sleep
    add      esp, 8
    push     64h ; dwMilliseconds
    call     api_Sleep
```

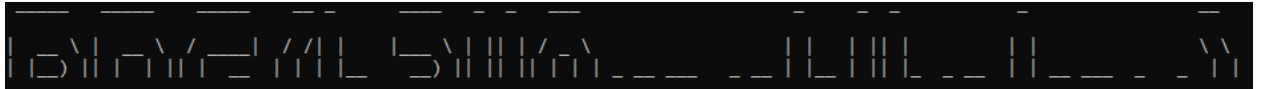
Перезапустив файл, получаем следующий вывод:



Референс ведет нас в главную функцию, из которой рисуется свинья и выполняются (предположительно) остальные проверки:

```
int sub_C81460()
{
    if ( Khabarovsk_function() )
    {
        if ( !sub_C811D0() )
        {
            print_pig();
            ExitProcess(0xFFFFFFFF);
        }
        sub_C810B0(&unk_C99E38, 117);
        sub_C810B0(&unk_C9A010, 117);
        sub_C81310();
    }
    return 0;
}
```

Патчим исполнение, приводящее к ExitProcess() и получаем следующий вывод:



Ага, значит необходимо найти и обойти все проверки, после чего, вероятно, мы получим флаг в ASCII арте.

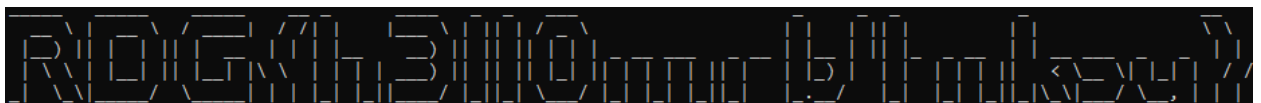
Немного потрассировав код, замечаем, что функция sub_C810B0 не что иное, как слегка модифицированный print. Переименовываем её в print_flag_part.

Рассмотрим функцию sub_C81310:

```
const CHAR *v0; // esi@1
unsigned int v1; // et@4
unsigned int v2; // et@4
HANDLE v8; // esi@8
HWND v9; // edi@9
HWND v10; // eax@9
DWORD BufferSize; // [sp+Ch] [bp-8h]@1
int v12; // [sp+10h] [bp-4h]@4

BufferSize = 4096;
v0 = (const CHAR *)LocalAlloc(0x40u, 0x1000u);
if ( !WNetGetProviderNameA(0x250000u, (LPSTR)v0, &BufferSize) )
{
    if ( lstrcmpiA(v0, "VirtualBox Shared Folders") )
        goto ExitProcess;
    print_flag_part((int)&unk_C998B0, 117);
}
v12 = 0;
v1 = __readeflags();
__writeeflags(v1 | 0x200000);
v2 = __readeflags();
if ( v2 & 0x200000 )
{
    _EAX = 1;
    __asm { cpushd }
    v12 = (_ECX & 0x80000000) != 0;
}
else
{
    v12 = 2;
}
if ( v12 != 1 )
    ExitProcess(3u);
print_flag_part((int)&unk_C9A3C0, 117);
v8 = CreateFileA("\\\\.\\pipe\\VBoxTrayIPC", 0x80000000, 3u, 0, 3u, 0, 0);
if ( v8 == (HANDLE)-1
    || (print_flag_part((int)&unk_C99A88, 117),
        CloseHandle(v8),
        v9 = FindWindowA("VBoxTrayToolWndClass", 0),
        v10 = FindWindowA(0, "VBoxTrayToolWnd"),
        !v9)
        && !v10 )
```

В этой функции (если немного погуглить) явно видны проверки на присутствие в виртуальной машине. Патчим их, получаем флаг:



Mobile (Nokia)

Abstract: Nokia is eternal. Such tasks too.

Гуглим списки тулз для стеги в wav файлах: <https://github.com/DominicBreuker/stego-toolkit>

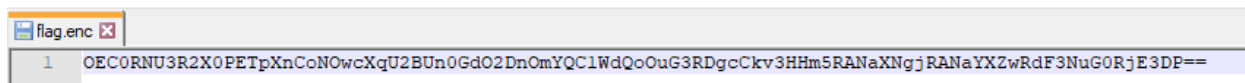
Смотрим, что там есть и выкидываем то, что использует пароли (как наиболее маловероятное)

Из вышеприведенного списка больше всего подходит stegpy, кормим ему наш аудиофайл и получаем флаг:

RDG{N0K1a_1s_3ternal}

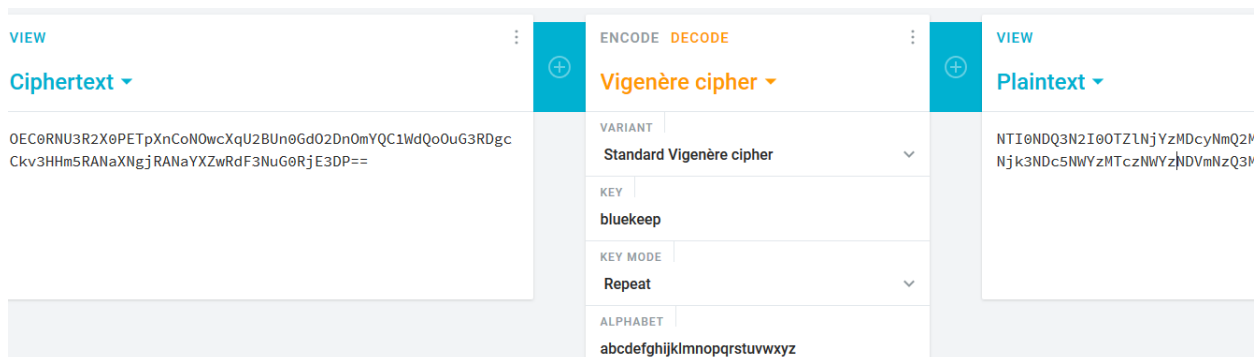
RLS

Abstract: Simple but effective. Some vulnerabilities are very dangerous. One of them was patched by MS in may 2019. Now live with it



Два знака == в конце дают намек на base64, однако содержимое на выходе некорректно -> можно сделать вывод, что изначально текст был зашифрован шифром замены.

На ум приходит шифр Виженера и шифр цезаря. Обращаемся к тексту задания и видим там подсказку на довольно известную уязвимость bluekeep:



Перегоняем текст из base64 в hex а затем в ascii:

5244477b496e6630726d6174316f6e5f53336375726974795f31735f345f747234707d ->

RDG{Inf0rmat1on S3curity 1s 4 tr4p}

BANK

Abstract: Banks, you know, like to use different exotic things. In our case, the whole system was created by one (!) very severe old coder. Deal with it now

Пробуем запустить – видим строку:

```
C:\Workdir\Writeps_RDG\task_5 (1)>crackme_5.exe
Usage: ./task <flag>

C:\Workdir\Writeps_RDG\task_5 (1)>_
```

Пробуем скормить бинарник IDA – видим дикий ад из совершенно непонятных конструкций:

base_GHCziBase_quotRemInt_info	.text	.text:0000000000402D90	var_70	= qword ptr -70h
base_GHCziBase_modInt_info	.text	.text:0000000000402D90	var_68	= qword ptr -68h
base_GHCziBase_divInt_info	.text	.text:0000000000402D90	var_60	= qword ptr -60h
base_GHCziBase_remInt_info	.text	.text:0000000000402D90	var_58	= qword ptr -58h
base_GHCziBase_quotInt_info	.text	.text:0000000000402D90	var_50	= qword ptr -50h
base_GHCziBase_getTag_info	.text	.text:0000000000402D90	var_48	= qword ptr -48h
base_GHCziBase_zdfMonadIO1_info	.text	.text:0000000000402D90	var_40	= qword ptr -40h
base_GHCziBase_bindIO_info	.text	.text:0000000000402D90	var_38	= qword ptr -38h
base_GHCziBase_zdfApplicativeIO2_info	.text	.text:0000000000402D90	var_30	= qword ptr -30h
base_GHCziBase_thenIO_info	.text	.text:0000000000402D90	var_28	= qword ptr -28h
base_GHCziBase_unIO1_info	.text	.text:0000000000402D90	var_20	= qword ptr -20h
base_GHCziBase_unIO_info	.text	.text:0000000000402D90	var_18	= qword ptr -18h
base_GHCziBase_zdfApplicativeIO4_info	.text	.text:0000000000402D90	var_10	= qword ptr -10h
base_GHCziBase_returnIO_info	.text	.text:0000000000402D90	var_8	= qword ptr -8
base_GHCziBase_zdfApplicativeIO3_info	.text	.text:0000000000402D90	argv	= dword ptr 10h
base_GHCziBase_liftAzuzdsiftA1_info	.text	.text:0000000000402D90		= qword ptr 18h
base_GHCziBase_ap1_info	.text	.text:0000000000402D91		
base_GHCziBase_apzudsap1_info	.text	.text:0000000000402D98		
base_GHCziBase_ap_info	.text	.text:0000000000402DA0		
base_GHCziBase_liftM1_info	.text	.text:0000000000402DA6		
base_GHCziBase_liftM2zuzdsiftM1_info	.text	.text:0000000000402DA0		
base_GHCziBase_liftM2_info	.text	.text:0000000000402DB2		
base_GHCziBase_zdfApplicativeOzuzdciftA2_info	.text	.text:0000000000402DB9		
base_GHCziBase_liftA1_info	.text	.text:0000000000402DBC		
base_GHCziBase_liftA3zuzdsiftA1_info	.text	.text:0000000000402DC0		
base_GHCziBase_zdfSemigroupIO2_info	.text	.text:0000000000402DC4		
base_GHCziBase_zdwzdsconcatA4_info	.text	.text:0000000000402DC8		
base_GHCziBase_zdfSemigroupIO1_info	.text	.text:0000000000402DCC		
base_GHCziBase_zdfSemigroupIO_info	.text	.text:0000000000402DD0		
base_GHCziBase_zdfSemigroupIOzuzdcstimes_info	.text	.text:0000000000402DD4		
		.text:0000000000402DD8		
		.text:0000000000402DDC		
		.text:0000000000402DE0		
		.text:0000000000402DE4		
		.text:0000000000402DE8		
		.text:0000000000402DEC		

push	rbp
sub	rsp, 100h
lea	rbp, [rsp+80h]
mov	[rbp+80h+argv], ecx
mov	[rbp+80h+argv], rdx
call	__main
mov	rax, cs:_refptr_defaultRtsConfig
mov	rdx, [rax]
mov	rcx, [rax+8]
mov	[rbp+80h+var_70], rdx
mov	[rbp+80h+var_68], rcx
mov	rdx, [rax+10h]
mov	rcx, [rax+18h]
mov	[rbp+80h+var_60], rdx
mov	[rbp+80h+var_58], rcx
mov	rdx, [rax+20h]
mov	rcx, [rax+28h]
mov	[rbp+80h+var_50], rdx
mov	[rbp+80h+var_48], rcx
mov	rdx, [rax+30h]
mov	rcx, [rax+38h]

Однако в названии функций можно заметить хинт на язык, на котором написан task, а немного порывшись в строках обнаружить следующие вещи:

.rdata:00000000...	0000000C	C	ьяяяяяяяpTs
.rdata:00000000...	0000000C	C	эяяяяяяяpTs
.rdata:00000000...	0000000C	C	юяяяяяяяpTs
.rdata:00000000...	00000018	C	SmvEc_c_qLNKKApxfAjNHZ
.rdata:00000000...	00000009	C	Congratz
.rdata:00000000...	00000015	C	Usage: ./task <flag>
.rdata:00000000...	00000017	C	Control.Exception.Base
.rdata:00000000...	00000011	C	PatternMatchFail
.rdata:00000000...	0000000C	C	RecSelError
.rdata:00000000...	0000000C	C	RecConError
.rdata:00000000...	0000000E	C	NoMethodError
.rdata:00000000...	0000000A	C	TypeError
.rdata:00000000...	00000009	C	<<loop>>
.rdata:00000000...	0000002D	C	Control.Concurrent.STM.atomically was nested
.rdata:00000000...	0000001B	C	Oops! Entered absent arg
.rdata:00000000...	00000011	C	in unboxed sum.
.rdata:00000000...	00000012	C	'PatternMatchFail

Единственный декомпилятор хаскеля работает только с ELF-файлами и для того чтобы получиться вменяемый листинг необходимо сначала пропатчить этот декомпилятор. Или потратить код и найти ужасно изуродованную функцию проверки

А можно просто прогуглить подобные задания и по ключу «Haskell + “Usage ./ ...”» найти ссылку на райтап по аналогичному таску и через него выйти на гит: https://github.com/Insomnihack/Teaser-2015/blob/master/Baby_Haskell/Baby_Haskell.hs

Шифр простой замены, подбираем наш флаг:

RDG{NANASKELL_tRlp_mEN}

SmackMyDumpUp

Abstract: Some fresh dumps from our trojans installed on administrative PCs. Enjoy

PCAP – дампы трафика. Кормим его любому анализатору (мне лень было ставить Wireshark поэтому я использовал онлайн-анализатор)

Connections	DNS	HTTP	SSL Certificates	PKI (X.509)	Transferred Files	Strange Activity	Similar Packet Captures			
<div><div><div>Q</div><div>Search in results</div></div></div>								<div><div>🔍</div><div>🔍</div><div>🔍</div></div>		
Timestamp	Connection ID	Sender IP	Sender Port	Target IP	Target Port	Transaction Depth	Method	Host	URI	Referrer
2020-03-10 12:23:19.2	C5oy11M9VvxEjci	192.168.14.148	51711	149.162.209.232	80	1	GET	www.tnyupload.com	/pgbar.php?/total=1&Read=0&Status=1&sessionid=prumd45jvrtbuqra467jp2t11	http://www.tnyupload.com/
2020-03-10 12:23:19.2	Ch6e1HXCOFZVUWj	192.168.14.148	51713	149.162.209.232	80	1	POST	www.tnyupload.com	/cgi-bin/upload.cgi?tsid=prumd45jvrtbuqra467jp2t11	http://www.tnyupload.com/
2020-03-10 12:23:19.2	C5oy11M9VvxEjci	192.168.14.148	51711	149.162.209.232	80	2	GET	www.tnyupload.com	/style.css	...ead=0&Status=1&sessionid=prumd45jvrtbuqra467jp2t11
2020-03-10 12:23:19.2	CmxE1h7Ch0BAxlc	192.168.14.148	51710	149.162.209.232	80	1	GET	www.tnyupload.com	...tbuqra467jp2t11&dnov=1503846599&dstart=1503846599	...ead=0&Status=1&sessionid=prumd45jvrtbuqra467jp2t11
2020-03-10 12:23:19.2	CLpJ3Q1xKEITovT70S	192.168.14.148	51710	149.162.209.232	80	1	GET	www.tnyupload.com	/favicon.ico	null
2020-03-10 12:23:20.2	Ch6e1HXCOFZVUWj	192.168.14.148	51713	149.162.209.232	80	2	GET	www.tnyupload.com	/file_uploaded_for_js.php?tsid=prumd45jvrtbuqra467jp2t11	http://www.tnyupload.com/
2020-03-10 12:23:23.2	CmxE1h7Ch0BAxlc	192.168.14.148	51710	149.162.209.232	80	2	GET	www.tnyupload.com	...tbuqra467jp2t11&dnov=1503846600&dstart=1503846599	null
2020-03-10 12:23:25.2	C254vH1oAkechYShb	192.168.14.148	51712	149.162.209.232	80	1	GET	www.tnyupload.com	/file_uploaded.php?file_id=0080562716631936170&del_id=09118122907934032354&gix=turning	http://www.tnyupload.com/
2020-03-10 12:23:25.2	C254vH1oAkechYShb	192.168.14.148	51712	149.162.209.232	80	2	GET	www.tnyupload.com	/images/flag_en.gif	...62716631936170&del_id=09118122907934032354&gix=turning
2020-03-10 12:23:25.2	ChSiX03ADcjZ6VYza	192.168.14.148	51722	149.162.209.232	80	1	GET	www.tnyupload.com	/images/flag_en.gif	...62716631936170&del_id=09118122907934032354&gix=turning

Заметив в списке URL ссылки на файлообменник проверяем список загрузок:

Connections	DNS	HTTP	SSL Certificates	PKI (X.509)	Transferred Files	Strange Activity	Similar Packet Captures				
<div><div>Q 149.202.220.122</div><div></div></div>								<div><div></div><div></div><div></div></div>			
Timestamp	Connection ID's	Artifact	MD5 Hash	SHA1 Hash	Originated From Host's	Sent To Host's	Source	Depth	Mime Type	File Name	Total Bytes
2020-03-10 13:23:19.2	Ch6e1HXCOFzVUWj	<div><div>73097801a127f96e</div><div></div></div>	<div><div>96363671d27029f96a0818a08029f96e</div><div></div></div>	<div><div>1585772013127106971067025087107025087</div><div></div></div>	<div><div>102.102.74.144</div><div></div></div>	<div><div>149.202.220.122</div><div></div></div>	HTTP	0	text/plain	<div><div>48.960-481.144</div><div></div></div>	null
2020-03-10 13:23:19.2	Ch6e1HXCOFzVUWj	<div><div>73097801a127f96e</div><div></div></div>	<div><div>96363671d27029f96a0818a08029f96e</div><div></div></div>	<div><div>1585772013127106971067025087107025087</div><div></div></div>	<div><div>102.102.74.144</div><div></div></div>	<div><div>149.202.220.122</div><div></div></div>	HTTP	0	text/plain	null	null
2020-03-10 13:23:19.2	Ch6e1HXCOFzVUWj	<div><div>73097801a127f96e</div><div></div></div>	<div><div>96363671d27029f96a0818a08029f96e</div><div></div></div>	<div><div>1585772013127106971067025087107025087</div><div></div></div>	<div><div>102.102.74.144</div><div></div></div>	<div><div>149.202.220.122</div><div></div></div>	HTTP	0	null	null	null
2020-03-10 13:23:19.2	C5oy111M09VkvEjcvl	<div><div>73097801a127f96e</div><div></div></div>	<div><div>96363671d27029f96a0818a08029f96e</div><div></div></div>	<div><div>1585772013127106971067025087107025087</div><div></div></div>	<div><div>149.202.220.122</div><div></div></div>	<div><div>102.102.74.144</div><div></div></div>	HTTP	0	text/html	null	null
2020-03-10 13:23:19.2	C5oy111M09VkvEjcvl	<div><div>73097801a127f96e</div><div></div></div>	<div><div>96363671d27029f96a0818a08029f96e</div><div></div></div>	<div><div>1585772013127106971067025087107025087</div><div></div></div>	<div><div>149.202.220.122</div><div></div></div>	<div><div>102.102.74.144</div><div></div></div>	HTTP	0	text/plain	null	null
2020-03-10 13:23:20.2	CrmvLEn31k7Ck0BAitc	<div><div>73097801a127f96e</div><div></div></div>	<div><div>96363671d27029f96a0818a08029f96e</div><div></div></div>	<div><div>1585772013127106971067025087107025087</div><div></div></div>	<div><div>149.202.220.122</div><div></div></div>	<div><div>102.102.74.144</div><div></div></div>	HTTP	0	text/html	null	null
2020-03-10 13:23:20.2	CLpJ3Q1nXEIT0v7705	<div><div>73097801a127f96e</div><div></div></div>	<div><div>96363671d27029f96a0818a08029f96e</div><div></div></div>	<div><div>1585772013127106971067025087107025087</div><div></div></div>	<div><div>149.202.220.122</div><div></div></div>	<div><div>102.102.74.144</div><div></div></div>	HTTP	0	text/html	null	294

Выкачиваем файл, получаем флаг:

RDG{Th1s_1s_very_3asy}

Итак. Приступим.

Greatest game

Всеми любимые герои 3. (сколько же я видел радости в глазах играющих)

На самом деле, всё было до ужаса просто – скачиваем файл greatest game, он оказывается архивом, внутри – HOMM3.

Запускаем игру любым возможным способом. Пытаемся начать игру, видим, что доступна всего одна карта. Заходим в игру. Бежим в центр карты – видим ящик пандоры – открываем.



Ящик говорит нам – “У тебя всё еще один герой?” (отсылает нас к любви большинства играть одним героем всю карту). Попробуй обменяться войсками и артефактами.

Проблем нет – слушаем совет коробки, берем второго героя – совершаем между ними обмен, видим –



Yara

```
$str = "f1139811bbf61362915958806ad30211"
```

condition:

```
uint16(0) == 0x5A4D and hash.md5(0, filesize) == $str and filesize < 200KB
```

Гугль говорит нам, что Yara – Это об описании файлов.

Ну мы просто берем и находим, от какого названия файла этот хэшг. Ez.

Fupercomputer

Получаем архив, в архиве 2 файла – encryptor и enc.

Encryptor на проверку оказывается python файлом. Enc – шифромясом.

Смотрим код:

Видим пару функций:

```
def is_prime(d, t):
    for i in range(t):
        rnd = randint(1, d - 1)
        if pow(rnd, (d - 1), d) != 1 :
            return False
    return True
```

Исходя из названия – это как-то проверяет число на простоту. Ок, допустим.

```
def gen_prime ():
    a = hashlib.md5(int.to_bytes(randint (1 << 60, 1 << 64), 10, 'big'))
    b = int.from_bytes(a.digest(), 'big') >> randint (114, 1 << 10)
    if b > 2 and b % 2 != 0 and is_prime(b, 1 << 16):
        r.add(b)
```

Генерирует простые числа соответственно.

```
with open ('flag', 'rb') as f:
    plain = int.from_bytes(f.read(), 'big')

while True:
    gen_prime()
    if len (r) == 1000:
        break
a = r.pop ()
b = r.pop ()
c = r.pop ()

x = a**b*c
#x = 71118419714714848277018497185854333397828600796

y = c ** b * a
with open ('enc', 'wb') as f:
    f.write (int.to_bytes(y - plain, 10000, 'big'))
```

Читает флаг – генерирует 1000 чисел. Берет последние 3.

Код на проверку является бутофорией с целью увеличения времени выполнения: какая нам разница – брать 1000 простых случайных чисел, или же взять 3?

Вернемся к функции генерации простых чисел. Пытаемся понять алгоритм.

```
a = hashlib.md5(int.to_bytes(randint (1 << 60, 1 << 64), 10, 'big'))
```

Хэш? Откуда тут хэш?? Ладно, смотрим дальше.

```
b = int.from_bytes(a.digest(), 'big') >> randint (114, 1 << 10)
```

Сдвиг вправо, а вот это уже интересно.

Этим сдвигом крайне большое число превращается в крайне маленькое. Путём небольшого эксперимента – и попытки уменьшить число проверок по `is_prime` - мы получим, что функция то отдаёт простые числа примерно до 16000

```
if b > 2 and b % 2 != 0 and is_prime(b, 1 < 16):
```

```
    r.add(b)
```

Остаётся мелочь:

Дано $x = a^{**}b^*c$

Найти $y = c^{**}b^*a$

Всё достаточно просто. Видим в конце большого числа

Наиболее простым является решение “в лоб” – перебираем все тройки простых чисел в раннее указанном диапазоне.

Даже не будем писать генератор простых чисел – воспользуемся данным.

```
old_r_len = 0
while True:
    gen_prime()
    if len(r) > old_r_len:
        old_r_len += 1
        print(old_r_len)
```

Как только цифорка перестанет быстро прибавляться – можно оставлять `r` в памяти – чисел будет скорее всего достаточно.

Напишем простейший код

```
>>> q = {}
>>> while True:
>>>     for i in r:
>>>         if x % i == 0:
>>>             x = x // i
>>>             if i in q:
>>>                 q[i] += 1
>>>             else:
>>>                 q[i] = 1
>>>             continue
>>>         if x == 1:
>>>             break
>>> q
{3: 10243, 5: 1}
```

Ой как интересно. А вот и числа. Расшифровываем.

(47 + 1) / 4 Ronin

Классика жанра. В описании видим “We intercepted a snapshot of a new cliché.”. Делаем вывод, что внутри скорее всего картинка. Какие мы знаем картинки? Png, jpg, gif

$(47 + 1) / 4 = 12$

Откуда тут 12.. Мммм... Ну попробуем посмотреть на структуру файла.

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text
00000000 8D BD 9E 8C 6C 69 2F 3C 2C 2D 65 78 73 64 60 40 [?Z?li/<,-exsd`@
00000010 6D 55 65 7A 9A 8A 65 5B 37 1D 08 0A 6C 79 28 37 mUezsSe[7...ly(7
00000020 65 41 65 79 72 6D 61 6D 6D 6B 65 79 65 6B 65 78 eAeyrmammkeyekex
00000030 72 64 61 6C 6C 79 65 7A 9A B0 65 3A 72 67 60 6D rdallyezs`e:rg`m
00000040 6E 78 64 78 67 69 67 7B 70 67 63 6F 69 7A 66 79 nxdxgig(pgcioizfy
00000050 66 68 63 7D 76 66 64 6B 6A 7E 62 7D 63 6C 62 71 fhc}vfdkj~b}clbq
00000060 7B 6E 68 64 64 73 6D 7D 62 61 68 73 78 6E 6D 60 (nhddsm)bahsxn`
00000070 60 75 62 73 6B 64 68 75 7C 6E 6D 60 60 86 BE 7A `ubskdhu|nm`~t*kz
00000080 26 6A 67 7B 70 66 62 6F 6A 7A 66 7C 69 63 62 71 &jg{pfbojzf|icbq
00000090 7E 69 6D 60 60 75 69 76 69 67 69 75 7E 69 6D 60 ~im`uivigiu~im`
000000A0 60 75 69 76 69 67 69 75 7E 69 6D 60 60 75 69 76 `uivigiu~im`uiv
000000B0 69 67 69 75 7E 69 6D 60 60 75 69 76 69 67 69 75 igiu~im`uivigiu
000000C0 7E 69 9E AC 6C 68 6D 7B F0 69 F1 7A 73 47 61 6E ~iz~lhm{ðifzsGan
000000D0 7D 78 66 6B 64 94 A1 79 6D 65 61 6D 69 78 64 7B }xfkd";ymeamixd{
000000E0 64 6A 64 79 72 65 61 6C 6C 79 65 7B 67 68 61 7C djdyzeallye{gha|
000000F0 74 62 69 65 66 72 9A BE 65 DE 75 79 70 64 62 6F tbiefrs%ePuypdbo
00000100 6E 7D 66 7F 60 6F 61 79 72 64 1C 6D 6E 7A 65 7E n)f.`oayrd.mnze~
00000110 74 6E 77 58 43 24 67 7F 3D 18 62 58 14 7F 57 F8 tñwXC$g.=bX..Wø
00000120 E3 C4 69 4F 2E C8 A4 6F 37 BA 95 5D 41 07 13 EE ããïO.Ê«7°•JÀ..i
00000130 65 73 73 6D 7D 72 7F 5C 54 42 49 45 46 4D 50 4C essm)r.\TBIEFMPL
00000140 52 53 5C 43 31 21 24 2A 2B 31 2C 30 36 3F 30 2F RS{C!|$*+1,06?0/
00000150 25 3D 38 36 0F 1D 00 1C 02 03 0C 13 01 11 14 1A %86.....
00000160 1B 01 1C 00 E6 EF E0 FF F5 ED E8 E6 FE EA F1 EF ....aiâyðieæpñi
00000170 F3 FC FD E0 E8 C7 C2 C8 C9 DF C2 D2 CC C1 D7 CA óúyáeÇÄÊËÄÖÏÀ×Ê
00000180 C6 D0 D7 DB D4 C0 DF B8 A6 AF A0 BF B5 AD A8 A6 ED×ÜÖÄß,|~µ.~|
00000190 BE AA B1 AF B3 BC BD A0 A8 84 83 8F 88 9C 83 9D %*±~³±s ~„f.œf.
000001A0 8D 82 8F 88 80 96 95 99 9A 8E 9D 83 9F 94 A1 79 .,.`E~³³Ž.fY";y
000001B0 6D 64 61 6F 6D 78 64 7B 64 6A 64 78 73 65 61 6C mdaomxd{djdxseal
000001C0 6C 79 65 7B 67 68 61 7C 74 62 69 65 66 72 9A BE lye{gha|tbiefrs%
000001D0 65 DE 74 79 70 64 63 68 68 7A 61 7D 60 6F 61 79 eBtypdchhza}`oay
000001E0 73 67 16 6C 6D 7B 66 6B 61 6E 44 48 74 77 20 3D sg.lm{fkanDhtw =
000001F0 6B 18 14 69 47 59 E4 71 66 27 F0 CD DD B8 6C 59 k..iGYäqf'ðÍY,1Y
00000200 56 39 95 6C 10 17 B0 66 7A 5D 51 9B 40 9A 72 61 V9•1..°fz]Q>@šra
00000210 6B 7F 47 4B 44 50 4F 4F 53 5C 5D 40 48 26 25 29 k.GKDPOOS\j@Hs%)
00000220 2A 3E 2D 33 2F 38 31 2C 24 32 39 35 36 1A 01 1F *>-3/81,$2956...
00000230 03 0C 0D 10 18 16 15 19 1A 0E 1D 03 1F E9 E6 FD .....éæý
00000240 F7 E3 E6 E4 E5 F3 F7 E9 F1 FE F3 EE EA FC FB CE -ãæãäó÷éñþóieüîf
00000250 CF DD C2 C3 CC D3 CC D6 D5 D9 DA CE DD C3 IYAUAAiOAOÖÜiYÄ
00000260 DF A9 A6 BD B7 A3 A6 A4 A5 B3 B7 A9 B1 BE B3 AE B@;~L;M¥³·@±%³@
00000270 AA BC BB BE EF 9D 80 9C 82 83 8C 93 80 96 95 99 ~æŽ..Eæ,fE"E~.³
00000280 9A 8E 9D 83 9F 94 BF 79 7E 66 60 6C 6E 68 66 6B šŽ.fY";y~f`lnhfk
00000290 65 54 65 84 8E C7 EB 44 6C DB EF 52 65 C9 EF 51 eTe„ŽÇeDlÜiReEiQ
```

Бывалому stfщику такая структура сразу скажет, что очень уж врятли это какой-то бличный, или иной модный шифр, ибо очень уж “бинарное мясо не мясное”.

Попробуем хог , тем более странная чиселка 12 прям нам намекает на это.

Классика – брутим хог ключ по заголовку длины 12. Особенно просто будет с jpg. У него то константный и самый частый заголовок именно 12 байт.

Как это сделать – описывать не буду. Всё крайне просто. Побайтовый ксор пользуясь свойством а хог b = c, с хог b = a, далее этим же ключем по всему файлу.

Шутка, приложу код, как бы я без него получил картинку...

```

jpg_header = [0xFF,0xD8,0xFF, 0xE0, 0x00, 0x10, 0x4A, 0x46, 0x49, 0x46,0x00, 0x01]

f = open ('47ronin','rb')

f12b = f.read (12)

key = []
for i in range(12):
    key.append (f12b[i] ^ jpg_header[i])

o = open ('47ronin.jpg','wb')
o.write (bytes(jpg_header))
while f12b:
    f12b = f.read (12)
    for i in range(len(f12b)):
        o.write (bytes([f12b[i] ^ key[i]]))

o.close()
|

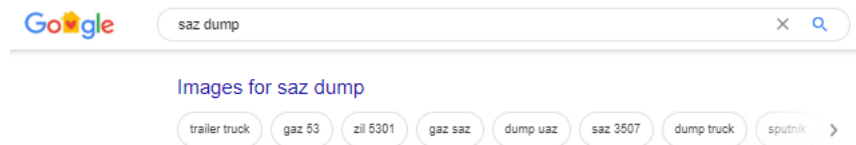
```

Вуаля..



Theatre

Открываем архив – видим файл sazdump. Думаем, что ж это такое. Гуглим



Видим.

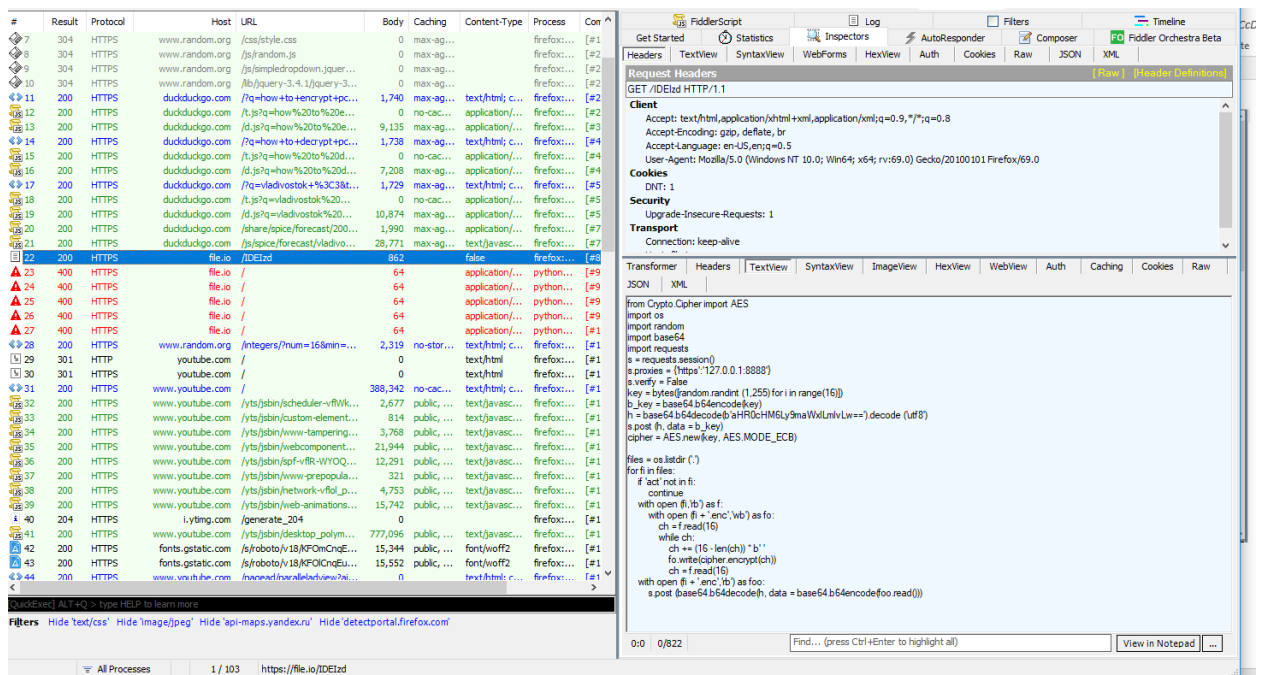
stackoverflow.com > questions > how-to-open-saz-webs...
How to open .saz websocket dump? - Stack Overflow
Apr 8, 2015 - The WebSocket file's format is not currently documented and direct manipulation is not supported. As of Fiddler 2.5.0.1, the format is as follows:
Can we speed up the dump command? - Stack Overflow Nov 15, 2019
How to parse data from Fiddler .saz file using python - Stack ... Jul 4, 2019
More results from stackoverflow.com

Ставим Fiddler, переименовываем файл. Открываем.

Видим кучу трафика.

#	Result	Protocol	Host	URL	Body	Caching	Content-Type	Process	Comments	Custom
1	200	HTTPS	www.fiddler2.com	/UpdateCheck.aspx?isBet...	785	private	text/plain; ...			
2	200	HTTPS	www.file.io	/	10,640		text/html	firefox:...	[#2]	
3	200	HTTPS	www.file.io	/assets/js/jquery.min.js	95,957		application/...	firefox:...	[#8]	
4	200	HTTPS	www.file.io	/assets/fineuploader-3.6...	64,907		application/...	firefox:...	[#9]	
5	200	HTTPS	www.file.io	/assets/fonts/fontaweso...	56,780		binary/octe...	firefox:...	[#13]	
6	200	HTTPS	www.random.org	/integers/?num=16&min=...	2,322	no-stor...	text/html; c...	firefox:...	[#17]	
7	304	HTTPS	www.random.org	/css/style.css	0	max-ag...		firefox:...	[#18]	
8	304	HTTPS	www.random.org	/js/random.js	0	max-ag...		firefox:...	[#22]	
9	304	HTTPS	www.random.org	/js/simpledropdown.jquer...	0	max-ag...		firefox:...	[#23]	
10	304	HTTPS	www.random.org	/lib/jquery-3.4.1/jquery-3...	0	max-ag...		firefox:...	[#24]	
11	200	HTTPS	duckduckgo.com	?q=how+to+encrypt+pc...	1,740	max-ag...	text/html; c...	firefox:...	[#28]	
12	200	HTTPS	duckduckgo.com	/t.js?q=how%20to%20e...	0	no-cac...	application/...	firefox:...	[#29]	
13	200	HTTPS	duckduckgo.com	/d.js?q=how%20to%20e...	9,135	max-ag...	application/...	firefox:...	[#35]	
14	200	HTTPS	duckduckgo.com	?q=how+to+decrypt+pc...	1,738	max-ag...	text/html; c...	firefox:...	[#47]	
15	200	HTTPS	duckduckgo.com	/t.js?q=how%20to%20d...	0	no-cac...	application/...	firefox:...	[#48]	
16	200	HTTPS	duckduckgo.com	/d.js?q=how%20to%20d...	7,208	max-ag...	application/...	firefox:...	[#49]	
17	200	HTTPS	duckduckgo.com	?q=vladivostok+%3C3&t...	1,729	max-ag...	text/html; c...	firefox:...	[#56]	
18	200	HTTPS	duckduckgo.com	/t.js?q=vladivostok%20...	0	no-cac...	application/...	firefox:...	[#57]	
19	200	HTTPS	duckduckgo.com	/d.js?q=vladivostok%20...	10,874	max-ag...	application/...	firefox:...	[#58]	
20	200	HTTPS	duckduckgo.com	/share/spice/forecast/200...	1,990	max-ag...	application/...	firefox:...	[#72]	
21	200	HTTPS	duckduckgo.com	/js/spice/forecast/vladivo...	28,771	max-ag...	text/javasc...	firefox:...	[#73]	
22	200	HTTPS	file.io	/IDEIzd	862		false	firefox:...	[#80]	
23	400	HTTPS	file.io	/	64		application/...	python...	[#96]	
24	400	HTTPS	file.io	/	64		application/...	python...	[#97]	
25	400	HTTPS	file.io	/	64		application/...	python...	[#98]	
26	400	HTTPS	file.io	/	64		application/...	python...	[#99]	
27	400	HTTPS	file.io	/	64		application/...	python...	[#100]	
28	200	HTTPS	www.random.org	/integers/?num=16&min=...	2,319	no-stor...	text/html; c...	firefox:...	[#103]	
29	301	HTTP	youtube.com	/	0		text/html	firefox:...	[#109]	
30	301	HTTPS	youtube.com	/	0		text/html	firefox:...	[#112]	
31	200	HTTPS	www.youtube.com	/	388,342	no-cac...	text/html; c...	firefox:...	[#115]	
32	200	HTTPS	www.youtube.com	/yts/jsbin/scheduler-vflWk...	2,677	public, ...	text/javasc...	firefox:...	[#118]	
33	200	HTTPS	www.youtube.com	/yts/jsbin/custom-element...	814	public, ...	text/javasc...	firefox:...	[#125]	
34	200	HTTPS	www.youtube.com	/yts/jsbin/www-tampering...	3,768	public, ...	text/javasc...	firefox:...	[#127]	
35	200	HTTPS	www.youtube.com	/yts/jsbin/webcomponent...	21,944	public, ...	text/javasc...	firefox:...	[#128]	
36	200	HTTPS	www.youtube.com	/yts/jsbin/spf-vflR-WYQO...	12,291	public, ...	text/javasc...	firefox:...	[#129]	
37	200	HTTPS	www.youtube.com	/yts/jsbin/www-prepopula...	321	public, ...	text/javasc...	firefox:...	[#130]	
38	200	HTTPS	www.youtube.com	/yts/jsbin/network-vflol_p...	4,753	public, ...	text/javasc...	firefox:...	[#131]	
39	200	HTTPS	www.youtube.com	/yts/jsbin/web-animations...	15,742	public, ...	text/javasc...	firefox:...	[#132]	

Ютуб конечно – это хорошо. Но что тут делает file.io



Мама родная.. Криптор. Что-ж он делает?

```
h = base64.b64decode(b'аHR0cHM6Ly9maWxlLmlvLw==').decode('utf8')
s.post(h, data = b_key)
Декодим.
```

Опять <https://file.io/> Странно.

В дампе видим 5 запросов, вернувших 400.

Первый явно ключ, остальные 4 – зашифрованные файлы из какой-то директории.

Пишем простой код.

```
import base64
import os
from Crypto.Cipher import AES

key = base64.b64decode('ZV1lcXKc5rTH6k460gPqdQ==')
cipher = AES.new(key, AES.MODE_ECB)

output = open('result', 'wb')
for i in range(1, 5):
    text = b''
    with open('file{}'.format(i)) as f:
        b64data = f.read()
        data = base64.b64decode(b64data)
    while data:
        b16 = data[:16]
        data = data[16:]
        output.write(cipher.decrypt(b16))
output.close()
```

Получаем текст. Открываем result. Ай да Джульета. А вот и флаг на 235й.

229 'Well, She's determined not to have a man.' **CR173**
230 **CR173**
231 'Oh God,' said Romeo. 'Such a waste. She's so beautiful, Benvolio. And
232 Benvolio grasped Romeo's wrist. 'Will you trust me? I can tell you how
233 **CR173**
234 'How?' said Romeo. 'Impossible. Tell me how.' **CR173**
235 {wh@t_is_th3_clty_but_th3_p30pl3} **CR173**
236 'Simple,' said Benvolio. 'Get out and about. Look at other girls.' **CR173**
237 **CR173**
238 'It's no good,' said Romeo. 'Whenever I see a beautiful girl from now
239 'I'm taking that as a challenge,' said Benvolio. 'I'll sort it out, do
other. And I don't think it's so difficult for men of our age.' **CR173**
240 **CR173**
241 His visitor, the young Count of Paris, shrugged. 'You're both respectful
this public brawling. But Capulet was rich and his daughter was very de
242 **CR173**
243 Capulet paced back and forth for a while, stopped and stared out the w:

Paparazzi

Задача немного посложнее ввиду своей структурной особенности.

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	4D	44	4D	50	93	A7	EE	A0	0F	00	00	00	20	00	00	00	MDMP^si
00000010	00	00	00	00	80	0A	66	5E	26	18	42	00	00	00	00	00e.f^s.B....
00000020	03	00	00	00	C4	00	00	00	60	06	00	00	11	00	00	00A...`.....
00000030	0C	01	00	00	24	07	00	00	04	00	00	00	34	0F	00	00S.....4...
00000040	30	08	00	00	09	00	00	00	90	0F	00	00	6E	B0	00	00	0.....n°..
00000050	10	00	00	00	D0	3F	00	00	9E	70	00	00	07	00	00	00B?..žp.....
00000060	38	00	00	00	D4	00	00	00	0F	00	00	00	54	05	00	00	8...ô.....T...
00000070	0C	01	00	00	0C	00	00	00	C8	1A	00	00	D6	55	00	00È...ÔU..
00000080	15	00	00	00	EC	01	00	00	64	17	00	00	16	00	00	00i...d.....
00000090	98	00	00	00	50	19	00	00	00	00	00	00	00	00	00	00	~...P.....
000000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000D0	00	00	00	00	09	00	06	00	03	4E	04	01	0A	00	00	00N.....
000000E0	00	00	00	00	EE	42	00	00	02	00	00	00	E8	19	00	00iB.....è...
000000F0	00	01	00	00	4C	77	F2	10	01	00	00	00	00	00	00	00Lwò.....
00000100	00	00	00	00	E8	00	29	00	D4	01	00	00	54	05	00	00è.)..ô...T...
00000110	F7	03	00	00	F0	1E	00	00	65	0A	66	5E	00	00	00	00	÷...ô...e.f^....
00000120	00	00	00	00	60	09	00	00	FC	08	00	00	E8	08	00	00`...ü...è...
00000130	03	00	00	00	03	00	00	00	00	20	00	00	0D	00	00	00
00000140	00	00	00	00	02	00	00	00	E0	01	00	00	50	00	61	00à...P.a.
00000150	63	00	69	00	66	00	69	00	63	00	20	00	53	00	74	00	c.i.f.i.c. .S.t.
00000160	61	00	6E	00	64	00	61	00	72	00	64	00	20	00	54	00	a.n.d.a.r.d. .T.
00000170	69	00	6D	00	65	00	00	00	00	00	00	00	00	00	00	00	i.m.e.....
00000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0B	00
00000190	00	00	01	00	02	00	00	00	00	00	00	00	00	00	00	00
000001A0	50	00	61	00	63	00	69	00	66	00	69	00	63	00	20	00	P.a.c.i.f.i.c. .
000001B0	44	00	61	00	79	00	6C	00	69	00	67	00	68	00	74	00	D.a.y.l.i.g.h.t.
000001C0	20	00	54	00	69	00	6D	00	65	00	00	00	00	00	00	00	.T.i.m.e.....
000001D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001E0	00	00	03	00	00	00	02	00	02	00	00	00	00	00	00	00
000001F0	C4	FF	FF	FF	31	00	37	00	31	00	33	00	34	00	2E	00	Àÿÿÿ1.7.1.3.4...
00000200	31	00	2E	00	61	00	6D	00	64	00	36	00	34	00	66	00	l...a.m.d.6.4.f.
00000210	72	00	65	00	2E	00	72	00	73	00	34	00	5F	00	72	00	r.e.e.r.s.4._r.
00000220	65	00	6C	00	65	00	61	00	73	00	65	00	2E	00	31	00	e.l.e.a.s.e...l.
00000230	38	00	30	00	34	00	31	00	30	00	2D	00	31	00	38	00	8.0.4.1.0.-1.8.
00000240	30	00	34	00	00	00	00	00	00	00	00	00	00	00	00	00	0.4.....

Джони, это же дамп!

Описание гласит:


Before we shot him, the damn photographer managed to take a picture of an extremely important object. The camera was broken and thrown into a trash can. Strange is its lens width: 960 ...

Фотографа то мы вообще застрелили, но точно знаем, что ширина кадра у него была 960. А это значит, что возможно перед нами дамп памяти фотоаппарата данного гражданина. А внутри Снимок с шириной 960. Что-ж. Классика.

Берем любой графический просмотрщик, который умеет смотреть raw картинки. Ну и открываем весь наш дамп условно бесконечной лентой по вертикали.

Выставляем ширину в 960. И наслаждаемся.

Я пользуюсь irfanview

Name	Date modified	Type	Size
 Paparazzi.raw	3/9/2020 2:21 AM	IrfanView RAW File	91,412 KB

Set RAW open parameters



File: [redacted]PaparazziPaparazzi.raw

Image width:

Image height:

File header size: bytes (will be skipped)

BitsPerPixel (BPP):

☐ 1 BPP

☐ Big endian

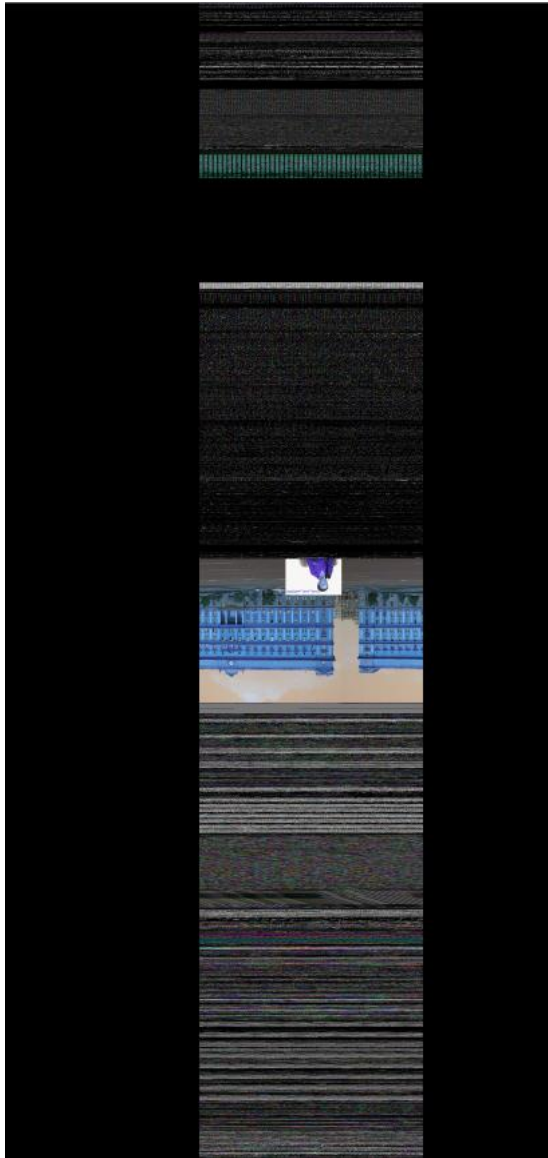
☐ 8 BPP (grayscale, 1 byte per pixel)

Misc.:

☐ Vertical flip

☐ Grayscale (for 16 or 24 BPP)

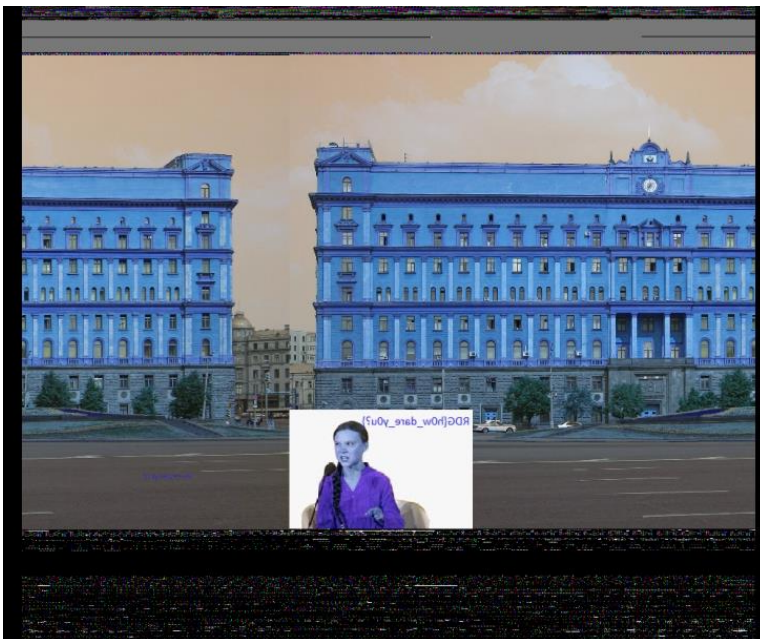
Видим



Странный участок посередине. Увеличиваем



Переворачиваем



Грета , лубянка, флаг, ы.

Ancient knowledges

Этот даже расписывать не буду. Открываем образ – крякаем через загрузчик root пароль – даем history – там будет флаг

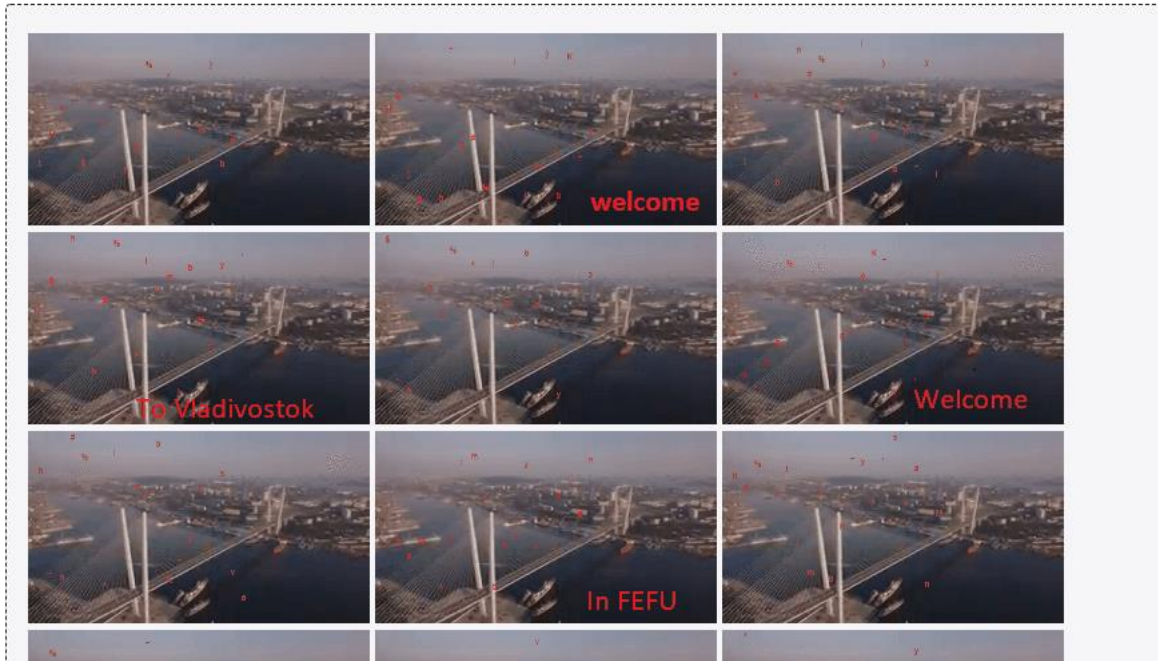
Broken heart

Rainy city

Бегает бегае бегае бегае гифка. Буковки бегают в ней.

У всех в команде был человек, который ничео не умел. Ну вот его задачей было искать тут флаг.

Split images:



Не, ну можно было применить CV. (hi kl'e'nin) Но руками явно быстрее флаг был на 228 фрейме.



Alldatalost и Site template

В архиве хатрр, открываем хатрр – пытаемся поднять веб сервер. Выясняется, что памяти на веб сервер явно не хватает.

Выделяем в конфиге память (а там всё занулено) – поднимаем хатрр – заходим в phpMyAdmin – получаем первый флаг.

Поднимаем единственный сайт, который там был доступен – открываем посты wp (предварительно сбрутив root root), в посте белым тестом второй флаг от site template.

Военная база

cHktZW5pZ21hDQpyb3RvcnM9J0lJIEkgSUIJJw0KcmVmbGVjdG9yPSdCJw0KcmZ19zZXRoZW5ncz1bMiwgMjlsIDhdDQpwbnHVnYm9hcmRfc2V0dGluZ3M9J0ZVIEhaIElOIETNIE9XIFJYIEFWIEJTIENHIERMJw0Kc2V0X2Rpc3BsYXkoJz8/PyсpDQonUFBYVEpCUUIZTExPRicNCg

Что превращается в

py-enigma

rotors='II I III'

reflector='B'

ring_settings=[2, 22, 8]

plugboard_settings='FU HZ IN KM OW RX AV BS CG DL'

set_display('???')

'PPXTJBQIYLLOF'

Качаем py-enigma – выставляем аналогичные настройки и брутим 3 по 27 позиций ротора – получаем флаг.

lottery

Читаем задание

Test yourself for luck. The program accepts an input parameter, it must be a number from 0 to 99, if you guess the desired number, you get a flag in the form of a created directory in the place where you run the file. Each launch number is different. Good luck.

Смотрим на файл, прикидываем, что шанс 1 к 100, дальше действуем исходя из знаний. Можно скриптом запустить зацикленный запуск файла с входным параметром, либо отревёрсить файл. Или если ты с иб, в тебе есть вагон целеустремлённости и телега усидчивости, тогда можно вручную запускать файл с входным параметром пока не повезёт.

madness

Как обычно читаем задание

do you like madness? me too. guess the name of the directory that I compiled.

Смотрим что у нас есть. Дан каталог с 9592 каталогами внутри. Заходим в него, смотрим на названия каталогов. Если математику ты не проспал, с лёгкостью увидишь, что это простые числа. названия каталогов лежат в диапазоне от 0 до 100 000. дальше остаётся самое простое, либо написать скрипт который получит список с названиями каталогов и проверит их на простоту, либо просто получаем список названий каталогов и проверяем в любом онлайн сервисе. Находим один каталог, который назван не простым числом.

Ay, Caramba

Таки читаем задание

We have a file server available to all residents of the Jankyville area, yesterday some hacker uploaded a strange file there. We made it inaccessible, but many people already managed to download it. You need to figure out what it is as quickly as possible, there is no time to hesitate.

Смотрим на файл, либо понимаем, либо нет, что дан exe файл собранный под .net на c#, который является обфусцированным ps1 скриптом. Запускаем, при запуске он проверяет производителя сри и сообщает всё хорошо или у тебя амд, попутно предлагая запустить файл с правами админа, запуск от админа вырубает сетевуху, надеемся что никто не запускает в своей ОС непонятные файлы еще и из под рута. для решения нужно открыть в дизассемблере, разобраться как он работает и найти инициализацию нужной переменной которая вычисляет флаг.

Для самых внимательных там по разным местам разбросаны около пяти отсылок к симпсонам.

io

Смотрим

"Comes in ... And comes out ... And comes in ... Great goes out!"

дана программа, пустой файл in и файл out с последовательностью символов , если записать в файл данные, при выполнении программы она перезапишет файл out новой последовательностью. несколько раз прогнав становится ОЧЕВИДНО что в out находится хеш. подставив в in любую строку и сравнив выход с выходом популярных хеш функций понимаем, что наша хеш функция sha1.

далее напрягаемся и вспоминаем, что sha1 не самая безопасная хеш функция, лезем в интернет и онлайн подборщиком быстро выясняем от чего был взят хеш, который лежал в Out.

الحاوية في الأزرق الحوت (svarog314)

Пытаемся прочесть задание

الحاوية في الأزرق الحوت (svarog314)

в описании сказано, о синем китее в контейнере. если вы не проспали последние 15 лет, то должны догадаться что речь идёт о докере. заходим в докер хаб, ищем там указанного пользователя. пулим себе контейнер, запускаем. обламываемся на ошибке про неверный таг. исправляем таг на стандартный latest, заходим в контейнер, видим mssql. дальше либо брутим пасс, либо ковыряем файлы. фалаг в названии базы.

nnn

Вот тут можно и не читать задание.

дана виртуальная машина. запускаем, пробегаемся по установленным программам, видим что стоит докер. заходим в службы, запускаем. запускаем сам докер, падаем в ошибку, думаем, еще думаем, вспоминаем/гуглим как работает докер под виндой, понимаем что он создаёт виртуальную машину, вспоминаем что мы уже в виртуальной машине, выключаем всё. включаем поддержку вложенной виртуализации. включаем всё, докер поднимается. проверяем контейнеры, там пусто. думаем, вспоминаем еще раз как работает докер под виндой, переключаемся в режим линукс контейнеров. смотрим список контейнеров, там тоже пусто. проверяем список образов, там есть один. получаем из него контейнер. запускаем. лучше сразу подумать и попробовать запустить с нужными ключами что бы провалиться в баш, получаем ошибку, баша нет. меняем ключи, стартуем, проваливаемся в консоль, можно там походить, узнать что стоит веб сервер. или можно сразу запустить с нужным ключом и увидеть какой порт пробросится. заходим на порт, видим поздравления и ключ.

infected

Читаем задание

To protect our server, we hired a student, and what came of it ... this is a collapse ... look what happened to the server, we have already fired the student.

Заходим на сервер, видим вин сервер, не видим окошечки. Сервер вырубается примерно через минуту.

всё это для того что бы кто то додумался заглянуть в список job,ов, можно и в список task,ов, но там есть лишние задания. Либо идём по долгому пути через журнал событий.

нужно отключить задание по выключению машины что бы не мешало. рядом лежит второй скрипт с названием readme, в котором даны три задания (задания простые, на выполнение действий в командной строке винды), в конце просят запустить exe, который проверит выполнены ли задания и выдаст ключ.

basic

При открытии ссылки на сайте была единственная информация:

`http://server/file.php?q=root`

Как можно было догадаться, при обращении к файлу `file.php` с параметром `q=root` что-то должно было произойти.

К сожалению, такого файла не существует. Если попробовать `index.php?q=root`, то скрипт выведет информацию о переменной `$_GET['q']`.

Есть параметр? Значит пробуем подобрать полезную нагрузку. Достаем из широких штанин SQLMap и указываем адрес сервера с эксплуатируемым параметром `q`.

Спустя небольшое время уязвимость найдена. Остается сделать дамп базы и вытащить оттуда флаг

signature

Задание идентично предыдущему за исключением того, что теперь в запросе хеш. По сценарию разработчик допустил уязвимость и хеш от запроса отдавался вместе с ответом.

Для получения хеша необходимо было отправить два параметра: `q` и `h`

При совпадении хеша, сгенерированного на стороне клиента и на стороне сервера запрос проходил в базу данных и исполнялся

Здесь необходимо было написать небольшой прокси-сервер или попробовать руками осуществить инъекцию, вытащив флаг. Наиболее простым был второй вариант.

i dont know

В современном вебе никуда без Composer. Менеджер пакетов (зависимостей) для PHP. Обычно зависимости выносят вне публичной директории, но разработчик оставил ее доступной для всех.

При открытии сайта необходимо было обнаружить папку `/vendor/`, внимательно осмотреть ее на наличие изначально существующих файлов. В файле `composer.json` при внимательном осмотре был пакет с автором `Flag Flagovich` и полным путем до файла `sqlite`. Далее нужно было скачать через браузер его и открыть любым текстовым редактором или редактором для `sqlite` и найти там флаг в одной из таблиц.