ROOTCTF 제 1회 서울디지텍고등학교 청소년 해킹방어대회 Write-Up

JunhoYeo(여준호) 10st / 3147p

MISC - Welcome(50)

```
1. 제 1회 서울디지텍고등학교 해킹방어대회
2. 에 오신 것을 환영합니다
3. 모든 문제의 정답은 다음과 같은 형식을 가지고 있습니다
4. 정답 형식 = FLAG{내용}
5.
6. FLAG{Welcome_to_Seoul_Digitech_ROOT_CTF}
```

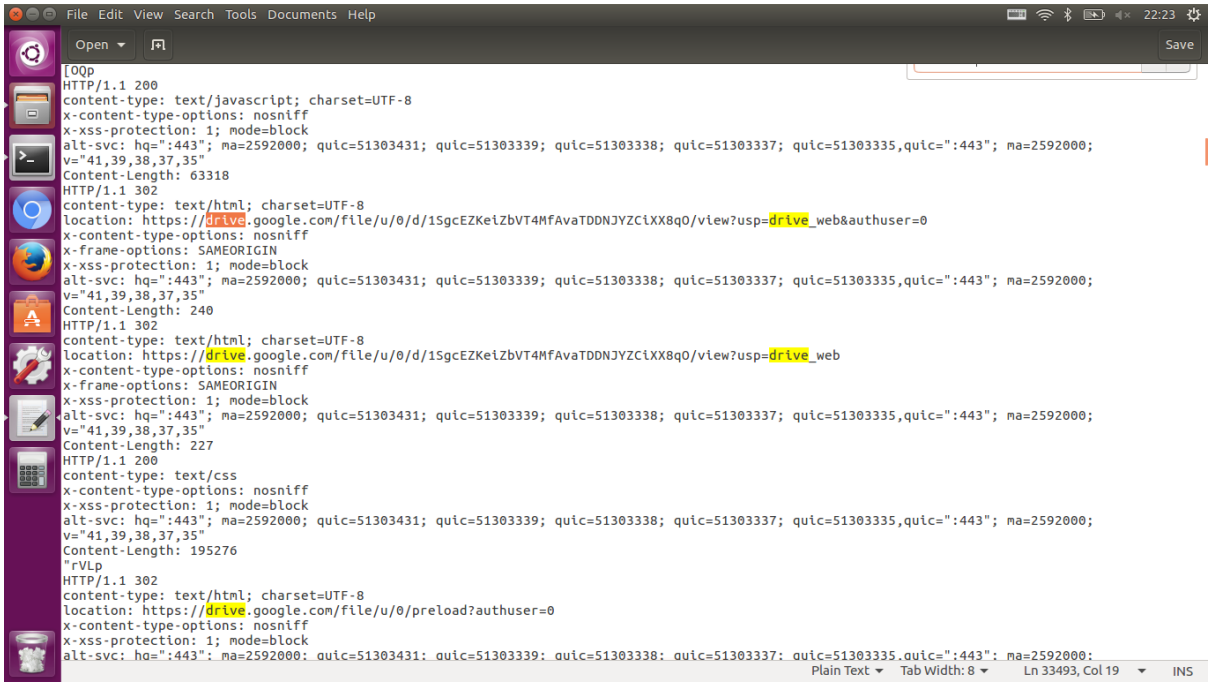
정답 형식을 알려주는 문제로, 그대로 문제에 나온 대로 입력하면 된다.

FLAG{Welcome_to_Seoul_Digitech_ROOT_CTF}

MISC - Find The Flag(913)

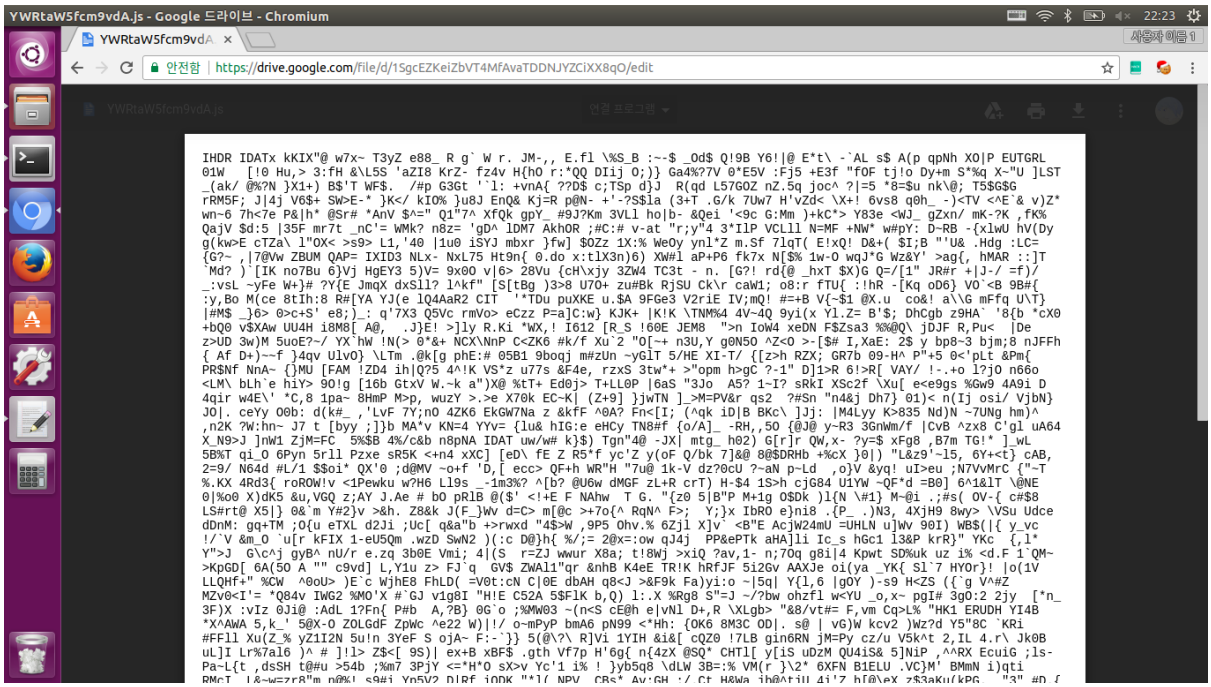
```
1. 문제 출제자는 크리스마스에 혼자 보내야 된다는 생각에 화가 나서 플래그를 숨겨버렸습니다.
2. 문제 출제자가 숨긴 플래그를 찾아주세요!
3. HINT:JS file,WebCacheV01.dat 분석
```

다른 문제들을 보면 알겠지만, 해당 CTF에서는 구글 드라이브를 통해 파일을 공유한다. 그렇기 때문에 JS file을 다운로드 받을 수 있는 구글 드라이브 공유 링크가 WebCacheV01.dat에 있을 것이라고 생각했다.

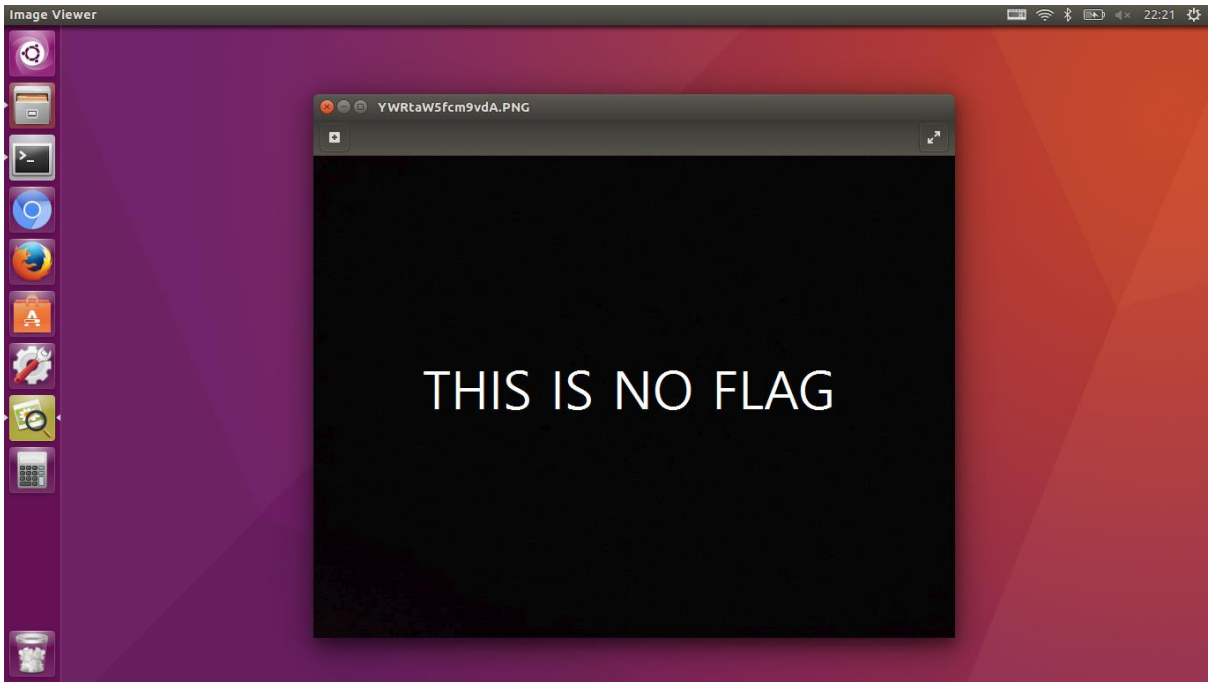


먼저 리눅스 터미널의 strings 명령어를 사용하여 WebCacheV01.dat에서 문자열 데이터만을 뽑아내

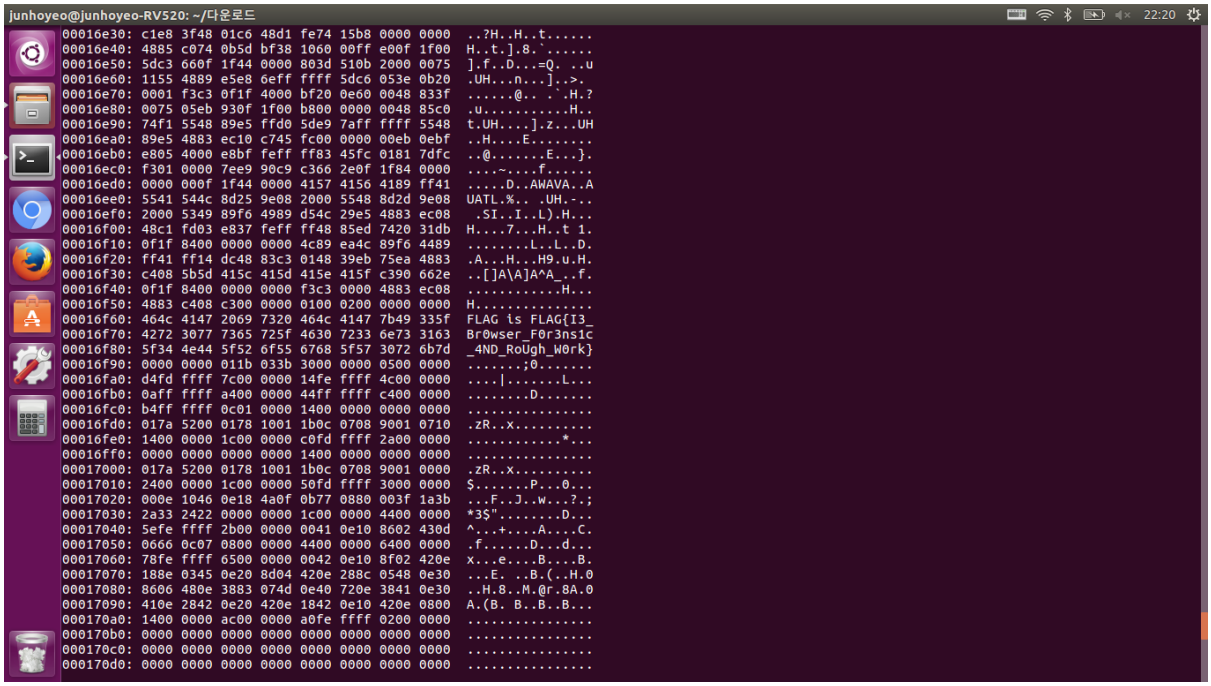
어 파일에 저장해 두었다. 그리고 검색 기능을 사용하여 구글 드라이브 공유 링크를 찾아냈다.



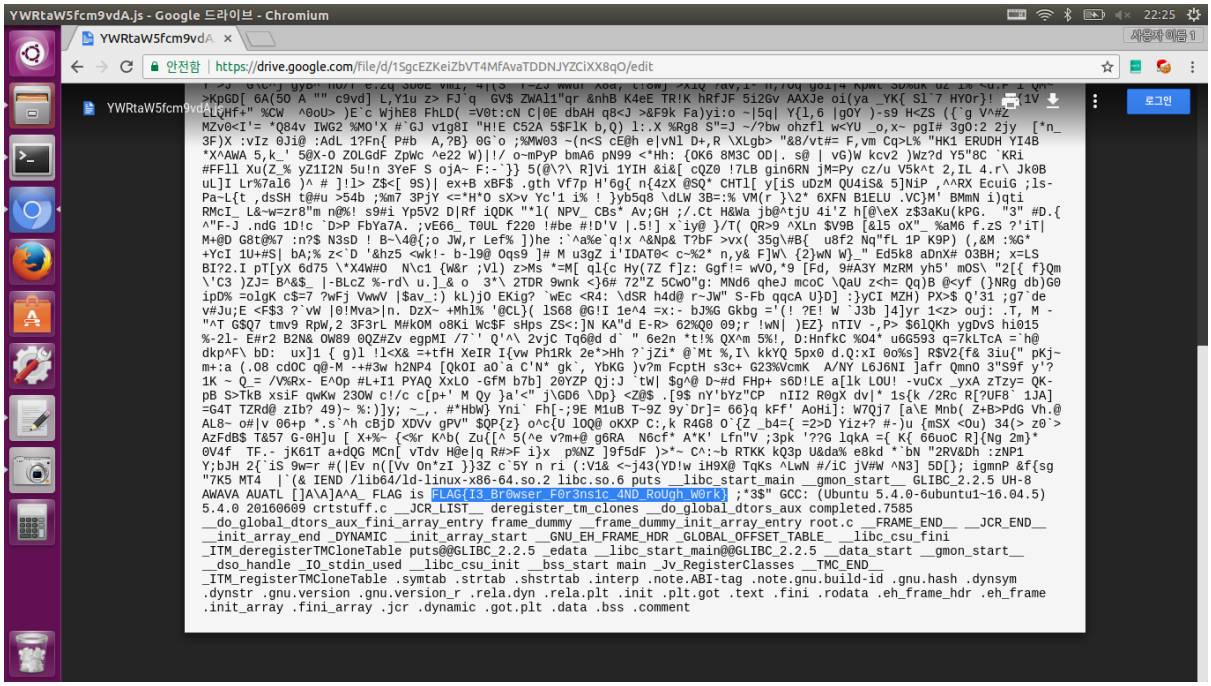
해당 링크에서 YWRtaW5fcm9vdA.js 파일을 다운로드할 수 있었다. 그러나 도저히 JavaScript file로는 보이지 않았다.



파일의 hex code를 보니 header signature가 PNG file로 되어 있어서, 확장자를 .PNG로 변경한 뒤 열어보았더니 플래그가 아니라고 떴다.



다행히 hex code 중간에서 플래그를 찾을 수 있었다.



여담이지만 그냥 다운받아서 hexdump 뜰 필요 없이 처음부터 플래그가 있었다... 쓥쓸

FLAG{I3_Br0wser_F0r3ns1c_4ND_RoUgh_W0rk}

MISC – Calculate(167)

```
1. 누가 내 패스워드좀 알려줘!
2. hint : 역연산
```

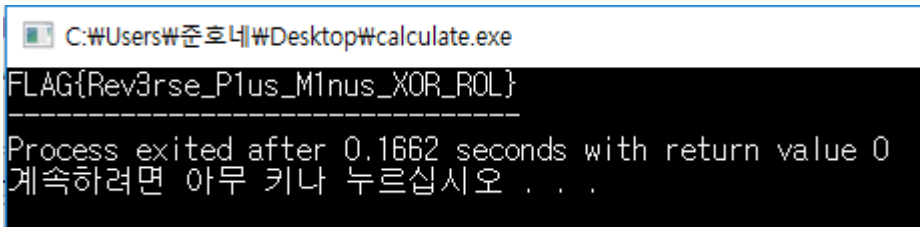
주어진 링크에 들어가니 Python으로 작성된 소스코드를 확인할 수 있었다.

jdoodle.com/python-programming-online 에서 코드를 실행하니 사용자에게 문자열을 입력받아 암호화 함수인 one(), two(), three(), four()를 순서대로 호출하여 문자열의 문자를 하나씩 암호화한 뒤 플래그값이 암호화되어 저장된 것으로 추정되는 result 배열의 값과 비교하여 일치하면 'Correct!!', 일치하지 않으면 'Incorrect..'를 출력하는 것 같았다.

```
1. #include <stdio.h>
2. int one(int num, int size){
3.     int r = num + size;
4.     r += 915;
5.     return r;
6. }
7. int two(int num, int size){
8.     int r = num - size;
9.     r -= 372;
10.    return r;
11. }
12. int three(int num, int size){
13.    int r = num ^ size;
14.    r ^= 826;
15.    return r;
16. }
17. int four(int num, int size){
18.    size %= 32;
19.    int r = num >> (32 - size);
20.    int b = (num << size) - (r << 32);
21.    return b + r;
22. }
23. int main(){
```

```
24.     int result[32] = {5040, 4944, 5088, 4992, 7232, 4848, 7584, 7344, 4288, 7408, 7
    360, 7584, 4608, 4880, 4320, 7328, 7360, 4608, 4896, 4320, 7472, 7328, 7360, 4608,
    4752, 4368, 4848, 4608, 4848, 4368, 4944, 7200};
25.     char data[100]="qwertyuiop{}asdfghjkl!?zxcvbnm,.QWERTYUIOPASDFGHJKLZXCVBNM_1234
    567890";
26.     for(int i=0; i<32; i++){
27.         for(int j=0; j<sizeof(data); j++){
28.             int number;
29.             int str=data[j];
30.             number=one(str,100);
31.             number=two(number,100);
32.             number=three(number,100);
33.             number=four(number,100);
34.             if(result[i]==number){
35.                 printf("%c", data[j]);
36.                 break;
37.             }
38.         }
39.     }
40.     return 0;
41. }
```

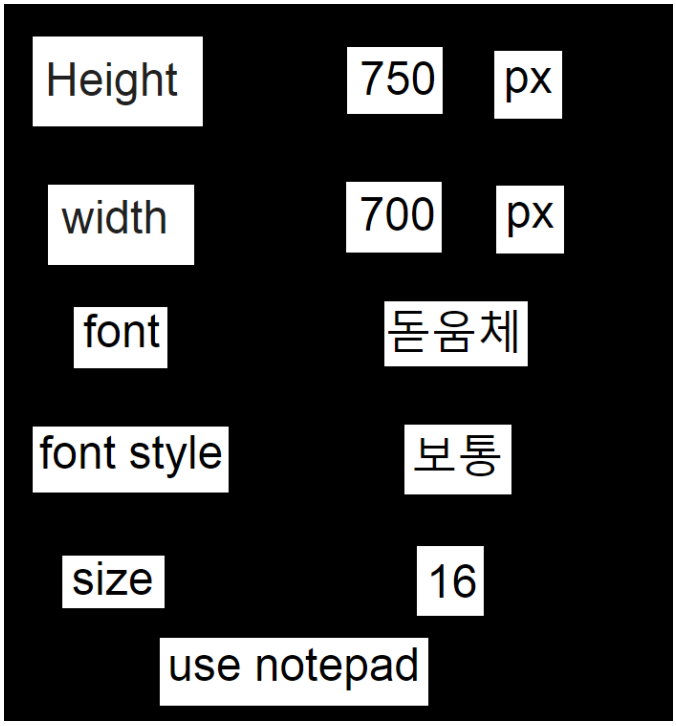
노가다 스피릿으로 하나씩 직접 입력해 코드표를 만드려는 생각도 들었지만 순간 이건 아니라는 것을 깨닫고 위와 같이 C 언어로 Flag 값을 출력하는 소스코드를 작성했다.
파이썬도 좋겠지만 C 언어로 하면 더 재미있을 것 같았기 때문에
-라고 쓰고 파이썬을 못해서 그랬다고 읽으면 될 것이다... 씨익
배열 result 에는 Flag 가 암호화된 값이, 배열 data 에는 Flag 를 이룰 것으로 추정되는 문자들이 저장되어 있다. 배열 result 에서 알 수 있듯이 Flag 는 총 32 개의 문자로 구성되어 있다.
이중 for 문을 사용하여 result[i]의 값과 data[j]를 암호화 함수를 순서대로 암호화한 값 number 를 비교해 두 값이 일치하면 해당 data[j]를 출력하고 break 하여 배열 result 의 다음 값을 구하고, 일치하지 않으면 배열 data 의 다음 값과 비교하는 구조로 플래그를 출력한다.
간단한 코드니까 금방 이해할 수 있을 거라고 생각한다.



프로그램을 실행하면 위처럼 Flag 가 나온다.
여담으로 다른 분들께서는 역연산 함수를 만들어서 푸신 분들이 많은 것 같다.
그런데 개인적으로는 저렇게 하나씩 암호화해서 비교하는 브루트 포싱으로 푸는 것도 재미있고 더 빨리 짤 수 있는 것 같다. 이는 역시 역연산 함수를 만드는 것을 못해서 그랬다고 읽으면 될 것 같다.
FLAG{Rev3rse_P1us_M1nus_XOR_ROL}

MISC – Vocabulary

1. 플래그가 적힌 친구의 단어장을 잃어버렸다
2. 어서 빨리 찾아야 된다.
3. 그 친구가 화내기 전에 플래그라도 찾아보자
4. hint : PNG height



pleas_find.png 파일을 다운로드하여 확인하고 청개구리처럼 hex editor로 열어봤는데 보기 불편해서 그냥 열라는 대로 notepad로 열어봤다.

```
-------------------------------------------------------
1장
vAluE 값                |vAry 서로 다르다              |
ExpEriENcE 경험         |ENSurE 반드시 .. 하게 하다    |
rEfErENcE 언급          |rEquESt 요청                  |
thrOugh ...을 통해      |vAry 서로 다르다              |
iDENtify 확인하다       |ENSurE 반드시 .. 하게 하다    |
cONcErN 영향을 미치다   |rEquESt 요청                  |
AblE 할 수 있는         |vAry 서로 다르다              |
likEly ...할 것 같은    |ENSurE 반드시 .. 하게 하다    |
-------------------------------------------------------
2장
rEMAiN 계속 ... 이다    |rEquESt 요청                  |
ENcOurAgE 격려하다      |vAry 서로 다르다              |
AppIE 사과              |ENSurE 반드시 .. 하게 하다    |
DEtErMiNE 알아내다      |rEquESt 요청                  |
iMpIEMENt 시행하다      |vAry 서로 다르다              |
NEcESSAry 필요한        |ENSurE 반드시 .. 하게 하다    |
gENErAl 일반적인        |rEquESt 요청                  |
-------------------------------------------------------
3장
ThiS lEttEr wAS firSt iNtrODucED iN ENglAND AND luckED
ArOuND thE wOrlD A yEAr, AND NOw thiS lEttEr tO yOu
ShOuld lEAvE yOu withiN fOur DAyS. YOu MuSt SEND SEvEN Of
thESE, iNcluDiNg thiS lEttEr, tO SOMEONE whO NEEDS gOOD
luck. COpyiNg iS AlSO rEcOMMENDED. It MAy bE
SupErStitiON, but it iS truE.IN ADDitiON, thE FLAG MAy
bE.....bEE.....bEEEE.....{ tHANk_FiND_MY_vOCAbuLAry
}. MAybE NOt. Or iNcrEASE the hEight tO 1000px.
```

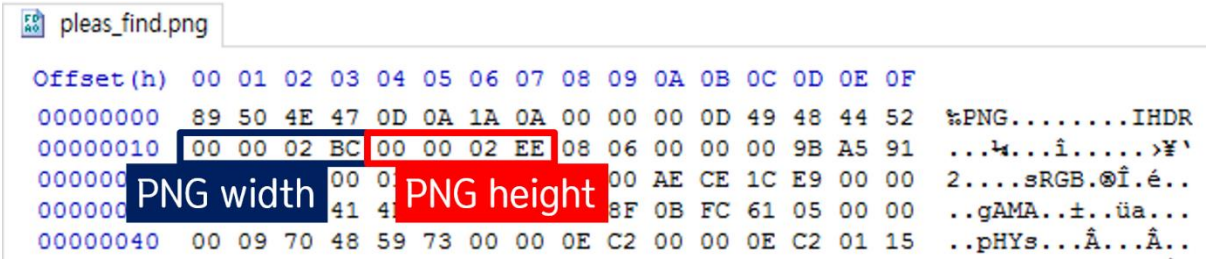
파일 끝부분에 단어장이 보인다. 영국에서 시작된 어디서 많이 본 듯한 편지와 Fake Flag, 그리고 height를 1000px로 increase하라는 힌트가 나왔다. 그래서 그림판을 사용해서 급한대로 옮겼더니 이미지가 깨져버렸다.

그렇다면 혹시 hex edit을 하여 PNG file의 height를 고치라는 게 아닐까?

물론 그걸 어떻게 할지 모르기 때문에! 구글링을 하여 자료를 찾아보았다.

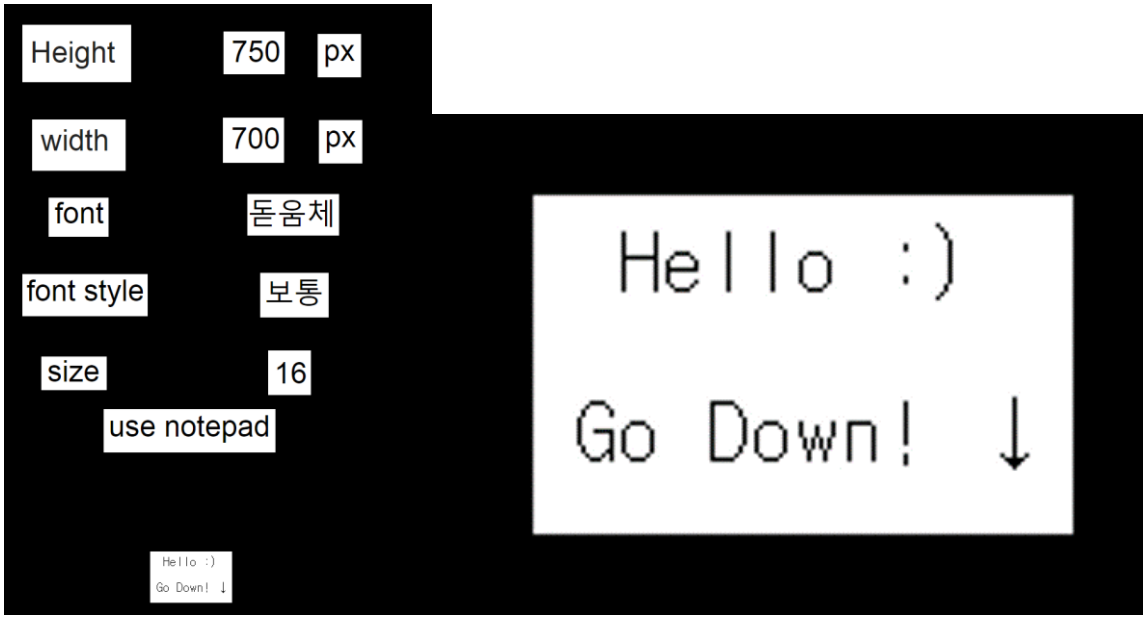
https://www.linkedin.com/pulse/hex-editing-width-height-png-files-ciaran-mc-ardle

PNG hex edit을 하여 이미지의 width, height data를 바꾸는 것에 관한 링크다.

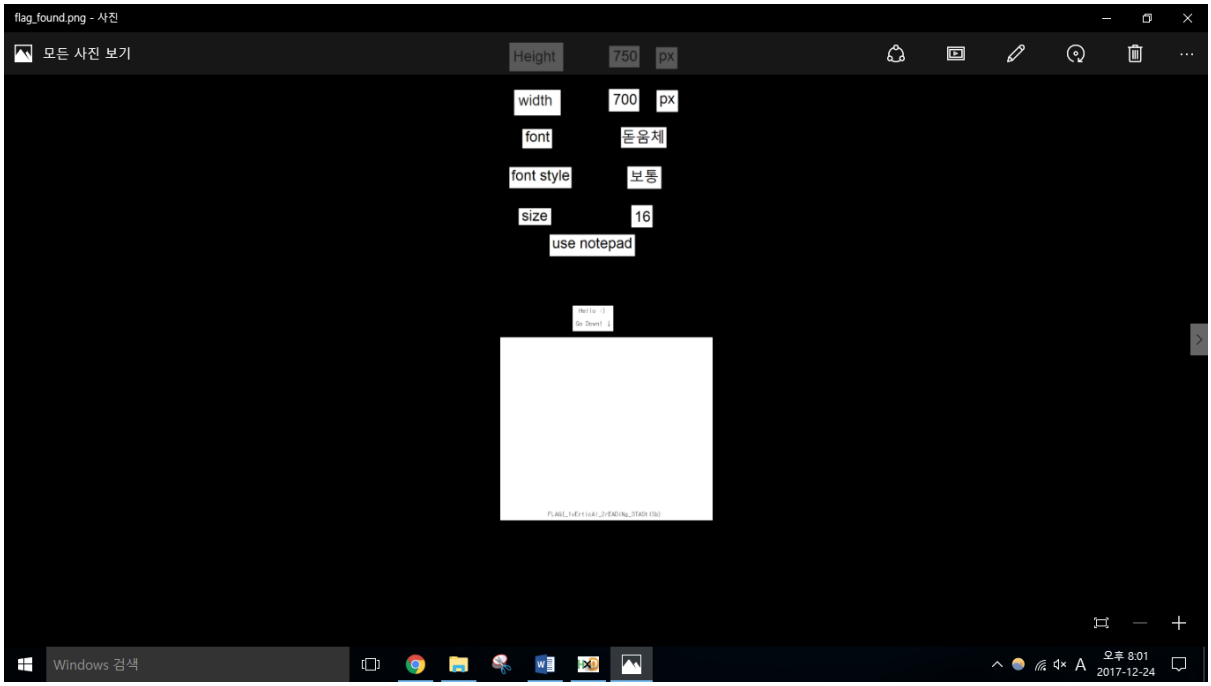


저기에 표시한 파란 부분을 수정하면 PNG width, 빨간 부분을 수정하면 PNG height를 고칠 수 있는 것 같다.

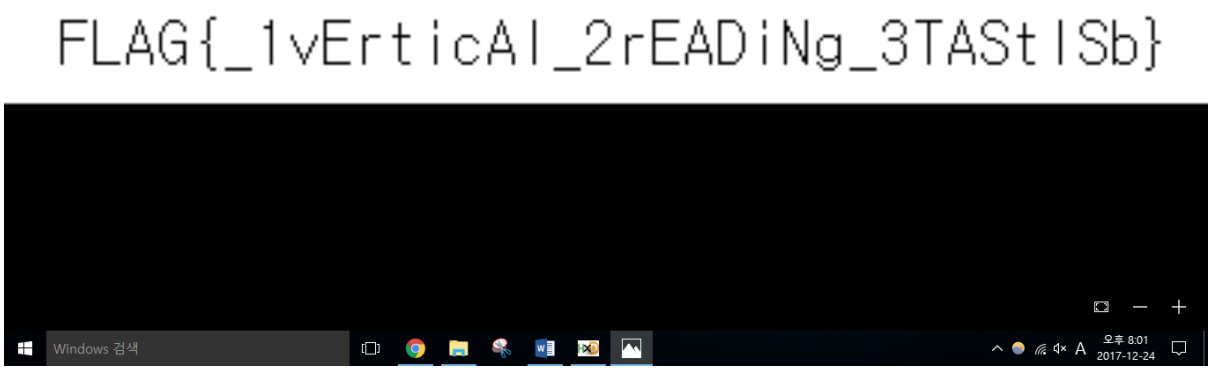
1000px로 바꾸라고 하였으므로 02 EE를 1000의 16진수 값인 03 E8로 고쳤다.



저 밑에 'Hello :) Go Down! ↓' 부분이 나타났다. 더 내려가면 플래그가 있는 듯하다. 그냥 두 배, 2000px으로 길이를 바꿔보기로 했다. 2000의 16진수 값인 07 D0으로 고쳤다.



어라라! 뭔가가 보인다.

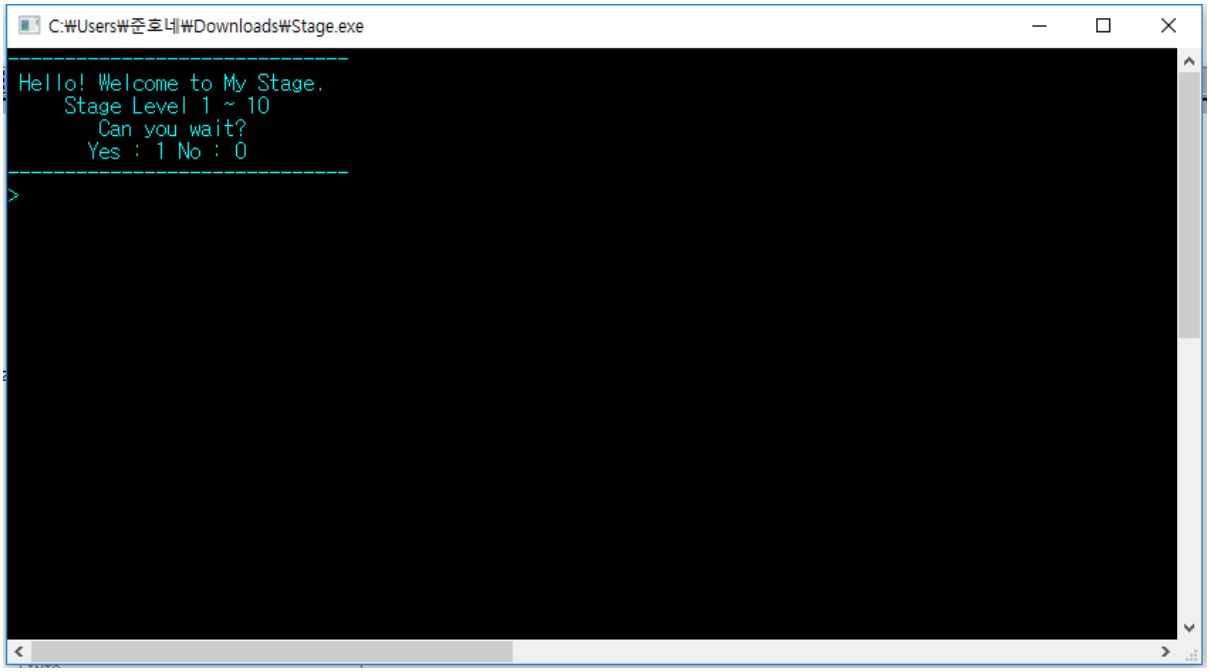


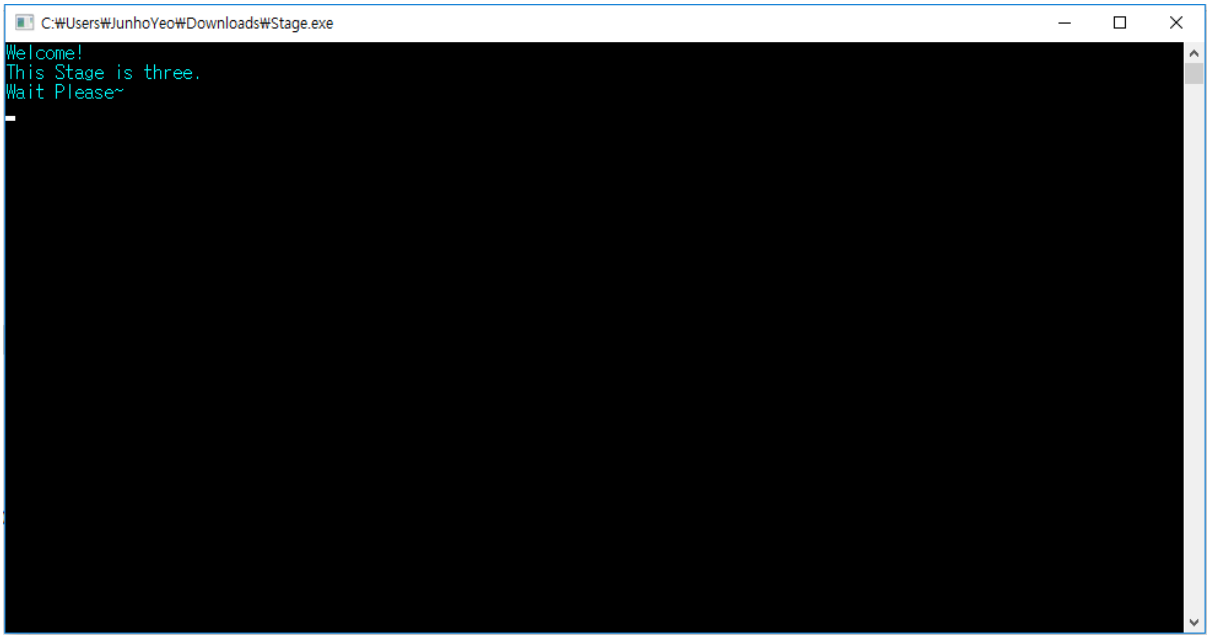
플래그가 나타났다!

FLAG{_1vErticAl_2rEADiNg_3TAStlSb}

REVERSING – Stage Game(229)

```
1. 인내의 시간..
2. Stage Level 1~10
3. hint : Sleep
```





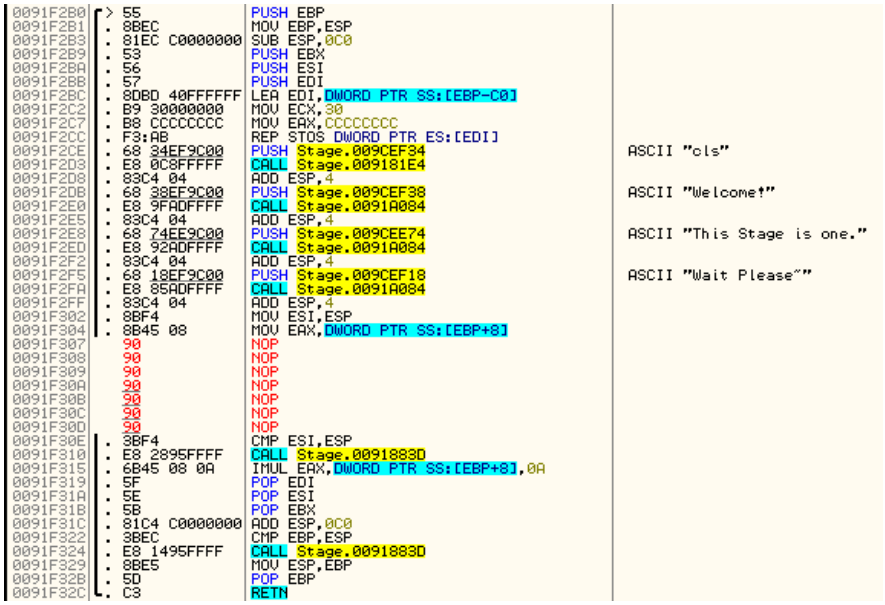
Stage.exe 파일이 주어진다. 실행해 본 결과 다음과 같은 프로그램인 것 같다. 먼저 실행되면 기다릴 수 있는지를 물어보고 1이면 스테이지 게임 시작, 0이면 종료를 한다. 게임 내용은 그냥 sleep() 함수 등으로 일정한 시간 동안 기다리게 하다가 다음 스테이지로 넘어가고 거기서 더 기다리면 다음 스테이지로 넘어가고 하는 것을 반복하면 플래그를 주고 끝나는 것 같다. 문제는 기다리는 시간이 너무 길다는 것이다. 본인의 인내심으로는 3번 스테이지가 한계인 것 같다... 리버싱해서 수정해 보자!

```
Address   Disassembly                                Text string
00917834  JMP Stage.0091FF60                         (Initial CPU selection)
0091EE8C  PUSH Stage.009CEEA8                        ASCII "color b"
0091EE99  PUSH Stage.009CF124                        ASCII "--------------------------------"
0091EEA6  PUSH Stage.009CEE54                        ASCII " Hello! Welcome to My Stage."
0091EEB3  PUSH Stage.009CEE88                        ASCII "      Stage Level 1 ~ 10 "
0091EEC0  PUSH Stage.009CEEB0                        ASCII "         Can you wait?    "
0091EECD  PUSH Stage.009CEED4                        ASCII "       Yes : 1 No : 0     "
0091EEDA  PUSH Stage.009CF124                        ASCII "--------------------------------"
0091EEE7  PUSH Stage.009CEEF4                        ASCII "> "
0091EEF8  PUSH Stage.009CEF14                        ASCII "%d"
0091EFE0  DD Stage.0091EFE4                          ASCII "check"
0091EFE4  ASCII "check",0
0091F00E  PUSH Stage.009CEF34                        ASCII "cls"
0091F01B  PUSH Stage.009CEF38                        ASCII "Welcome!"
0091F028  PUSH Stage.009CEFF0                        ASCII "This Stage is nine."
0091F035  PUSH Stage.009CEF18                        ASCII "Wait Please~"
0091F2CE  PUSH Stage.009CEF34                        ASCII "cls"
0091F2DB  PUSH Stage.009CEF38                        ASCII "Welcome!"
0091F2E8  PUSH Stage.009CEE74                        ASCII "This Stage is one."
0091F2F5  PUSH Stage.009CEF18                        ASCII "Wait Please~"
0091F36E  PUSH Stage.009CEF34                        ASCII "cls"
0091F37B  PUSH Stage.009CF004                        ASCII "Welcome to Final Stage!!"
0091F388  PUSH Stage.009CF020                        ASCII "This Stage is ten."
0091F395  PUSH Stage.009CF040                        ASCII "Wait Please"
0091F40E  PUSH Stage.009CEF34                        ASCII "cls"
0091F41B  PUSH Stage.009CEF38                        ASCII "Welcome!"
0091F428  PUSH Stage.009CEEF8                        ASCII "This Stage is two."
0091F435  PUSH Stage.009CEF18                        ASCII "Wait Please~"
0091F4AE  PUSH Stage.009CEF34                        ASCII "cls"
0091F4BB  PUSH Stage.009CEF38                        ASCII "Welcome!"
0091F4C8  PUSH Stage.009CEF5C                        ASCII "This Stage is three."
0091F4D5  PUSH Stage.009CEF18                        ASCII "Wait Please~"
0091F54E  PUSH Stage.009CEF34                        ASCII "cls"
0091F55B  PUSH Stage.009CEF38                        ASCII "Welcome!"
0091F575  PUSH Stage.009CEF18                        ASCII "Wait Please~"
0091F5EE  PUSH Stage.009CEF34                        ASCII "cls"
0091F5FB  PUSH Stage.009CEF38                        ASCII "Welcome!"
0091F608  PUSH Stage.009CEF8C                        ASCII "This Stage is five."
0091F615  PUSH Stage.009CEF18                        ASCII "Wait Please~"
0091F68E  PUSH Stage.009CEF34                        ASCII "cls"
0091F69B  PUSH Stage.009CEF38                        ASCII "Welcome!"
0091F6A8  PUSH Stage.009CEFA8                        ASCII "This Stage is six."
0091F6B5  PUSH Stage.009CEF18                        ASCII "Wait Please~"
0091F72E  PUSH Stage.009CEF34                        ASCII "cls"
0091F73B  PUSH Stage.009CEF38                        ASCII "Welcome!"
0091F748  PUSH Stage.009CEFC0                        ASCII "This Stage is seven."
0091F755  PUSH Stage.009CEF18                        ASCII "Wait Please~"
0091F7CE  PUSH Stage.009CEF34                        ASCII "cls"
0091F7DB  PUSH Stage.009CEF38                        ASCII "Welcome!"
0091F7E8  PUSH Stage.009CEFD8                        ASCII "This Stage is eight."
0091F7F5  PUSH Stage.009CEF18                        ASCII "Wait Please~"
```

올리디버거로 열고 All referenced text strings으로 문자열만 모아서 확인해보니 저렇게 문자열들이 나온다. ~~아아 귀분좋아~~

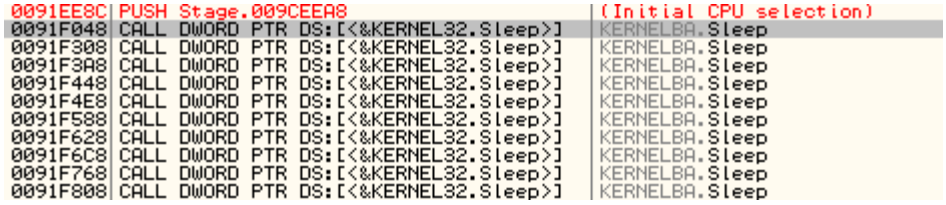
```
0091F2B0  > 55              PUSH EBP
0091F2B1  . 8BEC            MOV EBP,ESP
0091F2B3  . 81EC C0000000   SUB ESP,0C0
0091F2B9  . 53              PUSH EBX
0091F2BA  . 56              PUSH ESI
0091F2BB  . 57              PUSH EDI
0091F2BC  . 8DBD 40FFFFFF  LEA EDI,DWORD PTR SS:[EBP-C0]
0091F2C2  . B9 30000000     MOV ECX,30
0091F2C7  . B8 CCCCCCCC     MOV EAX,CCCCCCCC
0091F2CC  . F3:AB           REP STOS DWORD PTR ES:[EDI]
0091F2CE  . 68 34EF9C00     PUSH Stage.009CEF34                  ASCII "cls"
0091F2D3  . E8 0C8FFFFF     CALL Stage.009181E4
0091F2D8  . 83C4 04         ADD ESP,4
0091F2DB  . 68 38EF9C00     PUSH Stage.009CEF38                  ASCII "Welcome!"
0091F2E0  . E8 9FADFFFF     CALL Stage.0091A084
0091F2E5  . 83C4 04         ADD ESP,4
0091F2E8  . 68 74EE9C00     PUSH Stage.009CEE74                  ASCII "This Stage is one."
0091F2ED  . E8 92ADFFFF     CALL Stage.0091A084
0091F2F2  . 83C4 04         ADD ESP,4
0091F2F5  . 68 18EF9C00     PUSH Stage.009CEF18                  ASCII "Wait Please~"
0091F2FA  . E8 85ADFFFF     CALL Stage.0091A084
0091F2FF  . 83C4 04         ADD ESP,4
0091F302  . 8BF4            MOV ESI,ESP
0091F304  . 8B45 08         MOV EAX,DWORD PTR SS:[EBP+8]
0091F307  . 50              PUSH EAX                             Timeout
0091F308  . FF15 AC919F00   CALL DWORD PTR DS:[<&KERNEL32.Sleep>]  Sleep
0091F30E  . 3BF4            CMP ESI,ESP
0091F310  . E8 2895FFFF     CALL Stage.0091883D
0091F315  . 6B45 08 0A      IMUL EAX,DWORD PTR SS:[EBP+8],0A
0091F319  . 5F              POP EDI
0091F31A  . 5E              POP ESI
0091F31B  . 5B              POP EBX
0091F31C  . 81C4 C0000000   ADD ESP,0C0
0091F322  . 3BEC            CMP EBP,ESP
0091F324  . E8 1495FFFF     CALL Stage.0091883D
0091F329  . 8BE5            MOV ESP,EBP
0091F32B  . 5D              POP EBP
0091F32C  . C3              RETN
```

저 중 하나를 클릭해 보면 역시! 예상대로 Sleep() 함수를 호출(call)하여 delay를 발생시킨다는 것을 볼 수 있다. 이걸 '아무것도 안 하는' 어셈블리 명령으로 바꾸면 될 것 같은데...

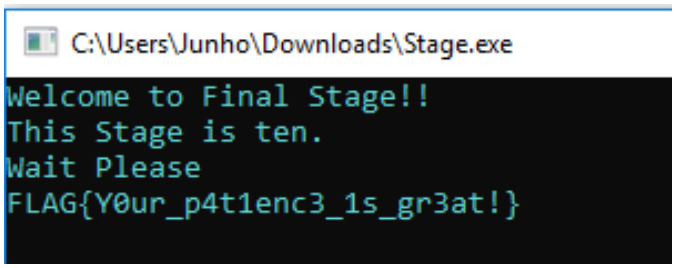


바로 NOP(No OPeration) 명령을 쓰면 될 것 같다. 어셈블리에서 아무 것도 안 하고 넘어가는 명령이다. 버퍼 오버플로우(BOF) 공격에서 NOP sled를 만들 때 사용하는 바로 그 명령!

저기 PUSH EAX 명령은 왠지 Sleep() 함수에 전달되는 인수인 것 같아서 그냥 CALL 명령이랑 같이 NOP처리를 했는데 왠지 안 했어도 됐을 것 같은 기분이 든다. 뭐 하는 명령인지도 궁금하다.



All intermodular calls로 사용한 API 함수들을 모아 볼 수 있다. 이걸 참고해서 다음으로 패치해야 하는 부분을 볼 수 있다. 또 마지막에 JMP로 Sleep() 함수로 넘어가는? 부분이 있던 것 같은데 그 부분도 패치해줘야 한다. 왜 그런지는 모르겠는데 아마 마지막 스테이지를 통과하고도 좀 더 기다려야 플래그가 나오는 게 아닐까? 리버싱 까막눈이라 모르겠지만 언젠가 똑! 보면 똑! 이해할 날이 오겠지...



저렇게 다 NOP로 패치한 프로그램을 실행하니 플래그가 나왔다. CTF에서 내가 처음으로 푼 리버싱 문제라서 그런지 정말 기분이 좋았다. 어떤 문서에서 쉘이 따지고 플래그가 나왔을 때의 쾌감이 있다는데 이런 게 바로 그런 것일까... 흐헹헹(?) ~~+)여기서는 처음이자 마지막 리버싱 문제였다칸다ㅠ~~

FLAG{Y0ur_p4t1enc3_1s_gr3at}

WEBHACKING – Login

```
1. 로그인 페이지인데 로그인이 안된다...
2. 로그인을 성공하고 짱해커가 되어보자!!
3. Hint : Array, length<6
4. Hint2 : Get으로 배열을 전송하는 방법, sql injection
```

진짜 오랫동안 헤맷지만 답은 꽤 가까이에 있었던 문제...

두개의 값, 하나의 변수

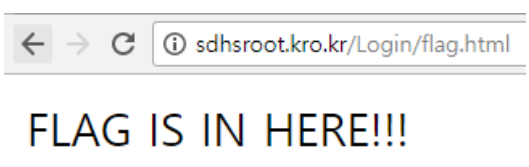
페이지에 들어가면 go! 버튼이 나오는데, 이를 누르면 login.php으로 pw=guest라는 값을 전달하며 리디렉션해준다.

```
<?php
include("dbcon.php");
$pw=$_GET['pw'];
$fpw=$_GET['pw'][1];
if(strlen($fpw)>5){
    echo "<script>alert('no hack~');location.href='login.html'</script>";
}
$query="select * from Login where pw='$fpw'";
$info=mysqli_query($con,$query);
$result=mysqli_fetch_array($info);
if($result['id']){
    setcookie("flag","VmxjeE1FNUdSbk5UV0hCclUwVmFiMWxzVm1GTlZtUnhVbFJXYVZKdGVGcFdSMOJYWWxaV1ZVMUVhejA9");
    echo "<script>location.href='flag.html'</script>";
}
highlight_file("login.php");
?>
```

login.php에 가면 위와 같은 PHP 소스코드를 볼 수 있다. _GET() 함수로 pw, fpw 변수의 값을 받아와서 SQL 어쩌고 저쩌고 해서 받아온 result 배열의 id값이 True면(?) flag라는 쿠키를 생성하고 저 값을 집어넣은 뒤 flag.html로 리디렉션 하는 것 같다. 그러면 flag.html에 Flag값이 나오겠지?

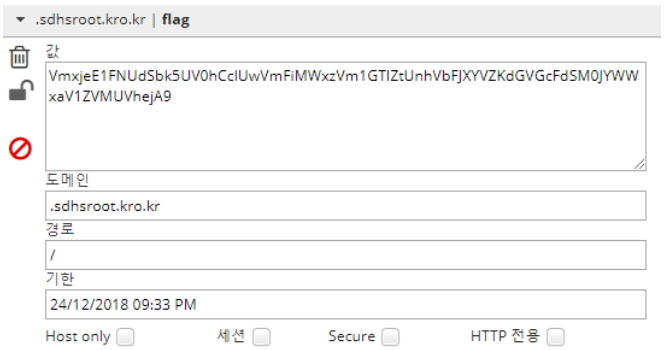
뭐 딱 봐도 SQL Injection(SQLI) 공격을 해야 하는 것 같다. 물론 본인은 SQL 지식이 전혀 없으므로 먼저 꼼수를 시도해봤다.

1) 그냥 flag.html로 고고싱



실패다!

2) 쿠키 값을 변조 후 flag.html로 고고싱



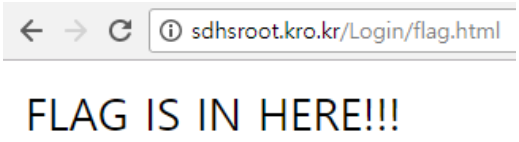
EditThisCookie라는 chrome extension을 사용해서 위와 같이 쿠키를 만들었다.

역시 FLAG IS IN HERE만 나올 뿐... 실패닷!

결국 SQL injection 뿐인가... 여러가지 방법들을 사용해 봤지만 리디렉션은 되지 않았다..

```
1.  http://sdhsroot.kro.kr/Login/login.php?pw[1]='OR'1
```

끈질긴 구글링 끝에 _GET()으로 array를 전달하는 방법을 겨우겨우 알아내 위처럼 SQL 인젝션을 시도했다.



Flag 쿠키가 생성되고 리디렉션은 되었지만... 플래그는 나오지 않았다. 페이지 소스보기를 해도 나오지 않는 시츄레이션이라 멘붕이 제대로 왔다.

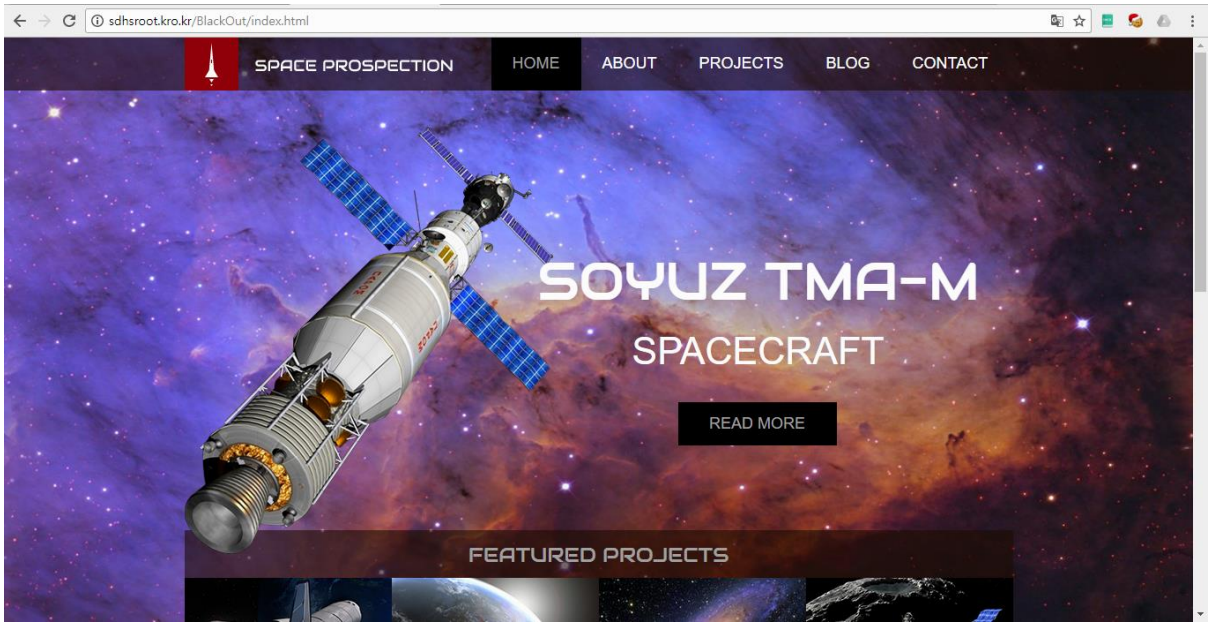
설마 Flag 쿠키에 저장되는 값이 플래그일까? Base64 format으로 decode를 시도해 보자 다시 Base64로 encode된 값이 나오길래 아래처럼 플래그가 나올 때까지 계속 decode하며 해결했다.

```
1.  VmxjeE1FNUdSbk5UV0hCclUwVmFiMWxzVm1GTlZtUnhVbFJXYVZKdGVGcFdSM0JYWWxaV1ZVMUVhejA9
2.  VlcxME5GRnNTWHBrU0Vab1lsVmFNVmRxUlRWaVJteFpWR3BXYlZWVU1Eaz0=
3.  VW10NFFsSXpkSEZoYlVaMVdqRTViRmxZVGpWbVVUMDk=
4.  Umt4QlIzdHFhbUZ1WjE5bFlYTjVmUT09
5.  RkxBR3tqamFuZ19lYXN5fQ==
6.  FLAG{jjang_easy}
```

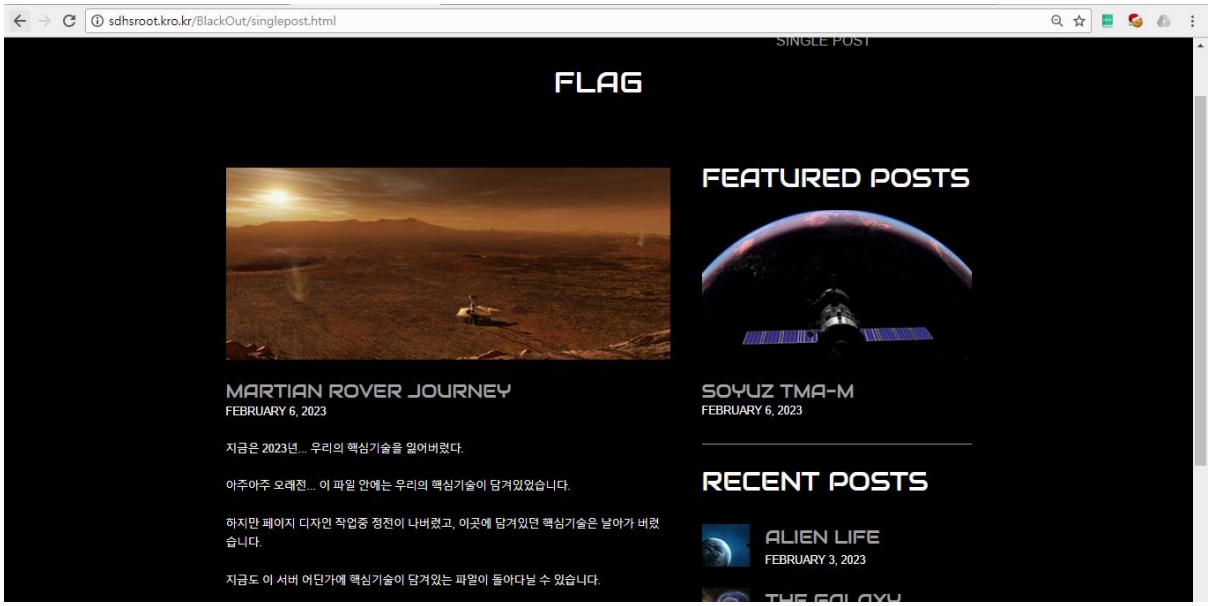
FLAG{jjang_easy}

WEBHACKING - SPACE PROSPECTION

```
1. 2023 년... SPACE PROSPECTION 라는 회사가 화성에 진출했다.
2. 회사의 사이트에 들어가 핵심 기술을 가져오자!!
```



주어진 링크인 sdhsroot.kro.kr/BlackOut/index.html에 접속했다.



사이트 여기저기를 뒤져보다가 BLOG 메뉴의 SINGLEPOST에 가니 위와 같은 Article이 나온다.

```
1. http://sdhsroot.kro.kr/BlackOut/singlepost.html
2. <h1>FLAG</h1>
3.     <div class="article">
4.     <img src="images/martianrover-journey.jpg" alt="">
5.     <h1>MARTIAN ROVER JOURNEY</h1>
6.     <span>FEBRUARY 6, 2023</span>
7.     <p>지금은 2023 년... 우리의 핵심기술을 잃어버렸다.</p>
```

```
8.      <p>아주아주 오래전... 이 파일 안에는 우리의 핵심기술이 담겨있었습니다.</p>
9.      <p>하지만 페이지 디자인 작업중 정전이 나버렸고, 이곳에 담겨있던 핵심기술은 날아가 버렸
    습니다.</p>
10.     <P>지금도 이 서버 어딘가에 핵심기술이 담겨있는 파일이 돌아다닐 수 있습니다.</P>
11. </div>
```

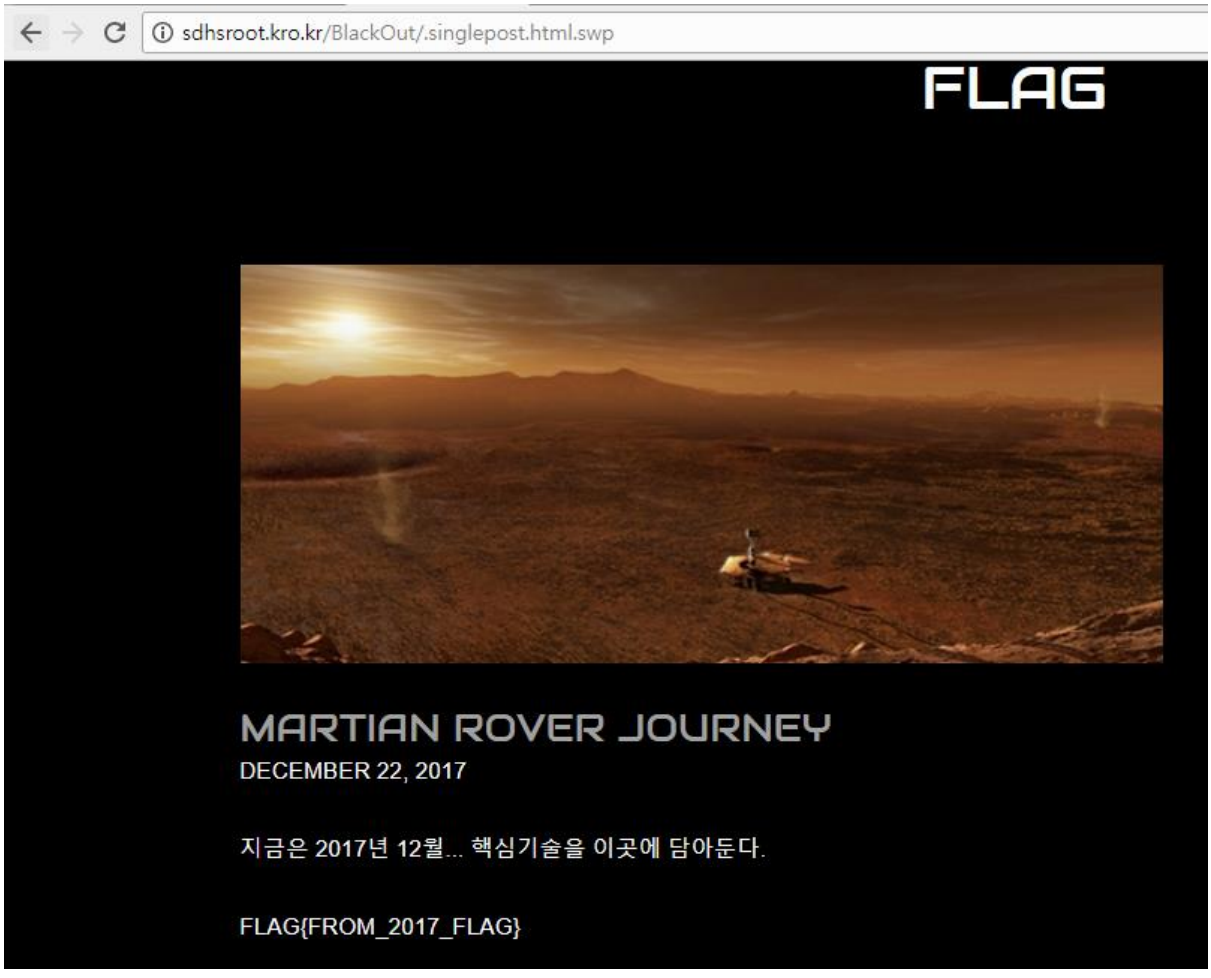
'이 파일'은 아마 Article에 있는 picture나 html file을 의미하는 것이라고 생각하고 다운받아 살펴보고 hex code도 확인하는 등 온갖 시도를 해보았고, 사이트에 있는 다른 파일들도 살펴보았다.

그러다가 '정전(blackout)'과 파일이 '날아갔다', '서버 어딘가에' 등으로 blackout 때문에 날아가버린 vi 에디터의 .swp 파일이 서버 어딘가에 남아있다는 것을 말해주려고 하는 것 같았다.

그렇다면 .swp 파일의 이름은 무엇일지 guessing해야 할 것이다. '.flag.txt.swp' 등으로 Flag와 관련하여 시도해보았지만 아니였다.

잠깐, '이 파일'이라고 했으므로 해당 html file과 관련되어 있지 않을까?

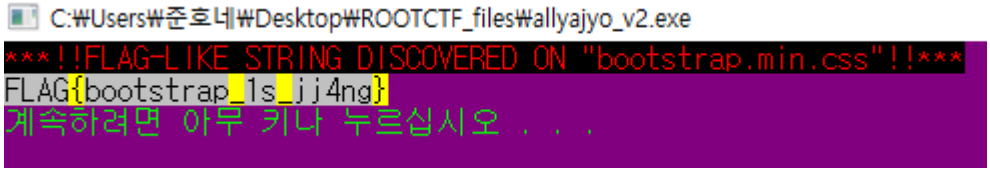
sdhsroot.kro.kr/BlackOut/.singlepost.html.swp에서 Flag를 찾을 수 있었다.



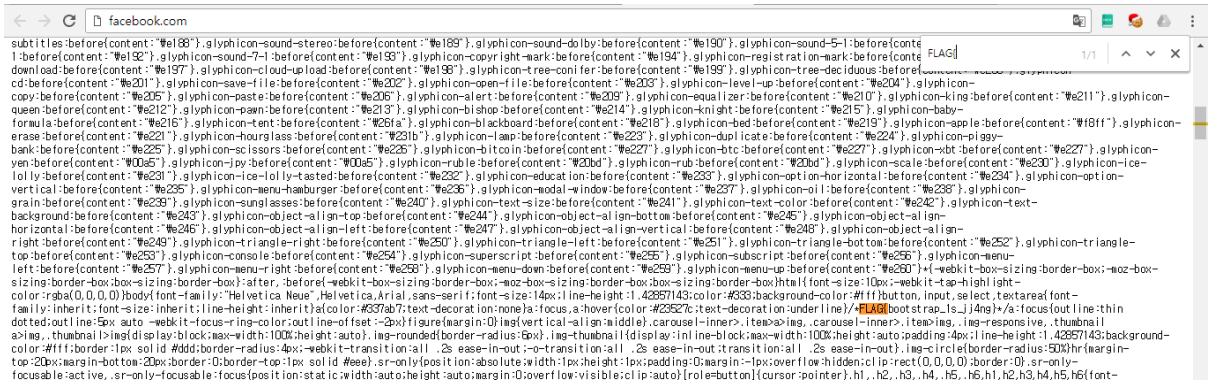
FLAG{FROM_2017_FLAG}

WEBHACKING - 보물찾기

이리저리 찾아보고 헤매다가 http://sdhsroot.kro.kr의 구성요소를 모두 다운받아 하나의 폴더에 저장하고, 개발 중이였던 해킹 툴의 소스코드를 약간 수정하여 실행했더니 Flag가 나왔다. ~~뭐 이런~~



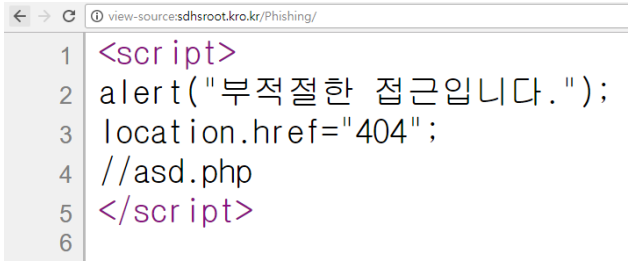
다른 CTF에서 파일의 hex dump에 Flag를 숨겨놓는 간단한 문제들이 나오는 것을 여러 번 보았기 때문에 Flag값을 더 빨리 찾아낼 수 있도록 재미 삼아 만든 툴이다. 파일의 hex code를 출력하고 Flag에 자주 쓰이는 키워드(Flag, FLAG, flag, ctf, CTF 등)를 hex code에서 검색해 출력하는 기능까지 구현했는데, 파일명을 하나 입력받아 해당 파일명의 파일만 검색하는 것에서 같은 디렉토리 안의 파일 전체를 대상으로 검색하고 hex code 출력 기능을 제거하여 사용했다. 보다시피 그냥 C언어로 쉽게 프로그래밍이 가능한 툴이다. 코드는 생략(잇힝)



저녀석에 따르면 http://sdhsroot.kro.kr/vendor/bootstrap/css/bootstrap.min.css의 주석 부분에 있다는데, 상상도 못한 곳이다. 깃허브 어쩌고길래 외부 모듈인 줄 알았는데. 문제 힌트에서 나는 모르는 무언가가 있었던 것일까.. 그냥 방심하지 말고 하나씩 차분하게 살펴보아야 하는 거구나...

WEBHACKING – Phishing

주어진 링크를 클릭하면 부적절한 접근이라는 알림이 표시되면서 이전 페이지로 리디렉션된다.



크롬에서 페이지 주소의 앞부분에 'view-source:'를 붙이자 위와 같이 페이지 소스를 볼 수 있었고, 주석으로 asd.php가 표시되어 있는 것 역시 확인이 가능했다.

sdhsroot.kro.kr/Phishing/asd.php에 접속하자 위처럼 FLAG{코드속에...}이라는 alart가 나온다. 물론 해당 Flag는 Fake로, 인증이 되지 않는다.

페이지 소스를 확인해 보니 난독화된 자바스크립트 코드가 있었다.

```
var b = 200;
for (a = 0; a <= 20; a++) {
    b = b + ((a * b) - (a / b));
    if (a == 0) b = 70;
    else if (a == 1) b = 76;
    else if (a == 3) b = 71;
    else if (a == 2) b = 65;
    else if (a == 4) b = 123;
    else if (a == 20) b = 125;
    else if (a == 5) {
        continue
    } else if (a == 6) {
        alert("코");
        continue
    } else if (a == 7) {
        alert("드");
        continue
    } else if (a == 8) {
        alert("속");
        continue
    } else if (a == 9) {
        alert("에");
        continue
    } else if (a == 10) {
        alert(".");
        continue
    } else if (a == 11) {
        alert(".");
        continue
    } else if (a == 12) {
        alert(".");
        continue
    } else if (a >= 4 && a <= 20) {
        continue
    }
    alert(String.fromCharCode(b))
}
```

jsbeautifier.org에서 난독화를 해제하자 위와 같은 코드가 나타났다.

```
<script>
var b = 200;
for (a = 0; a <= 20; a++) {
    b = b + ((a * b) - (a / b));
    if (a == 0) b = 70;
    else if (a == 1) b = 76;
    else if (a == 3) b = 71;
    else if (a == 2) b = 65;
    else if (a == 4) b = 123;
    else if (a == 20) b = 125;
    document.write(String.fromCharCode(b))
}
</script>
```

위와 같이 코드를 수정한 뒤 js.do에서 실행하자 문제의 힌트에서처럼 깨진 문자열로 이루어진 Flag가 나왔다.

FLAG{'ᐅ\⚓겣뻴豳郲놓볶□ꝯ□ɑ̧ㆬ□}

FLAG{'ᐅ\⚓겣뻴豳郲놓볶□ꝯ ɑ̧ㆬ }