ARGOS Hacking Festival 2011

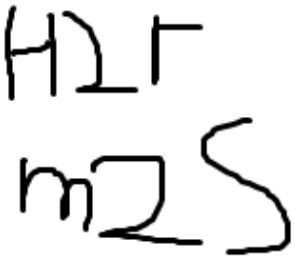
Write-Up – H2rm2s

Written By Xero

박준혁 (한국디지털미디어고등학교 1학년)

2011-11-28

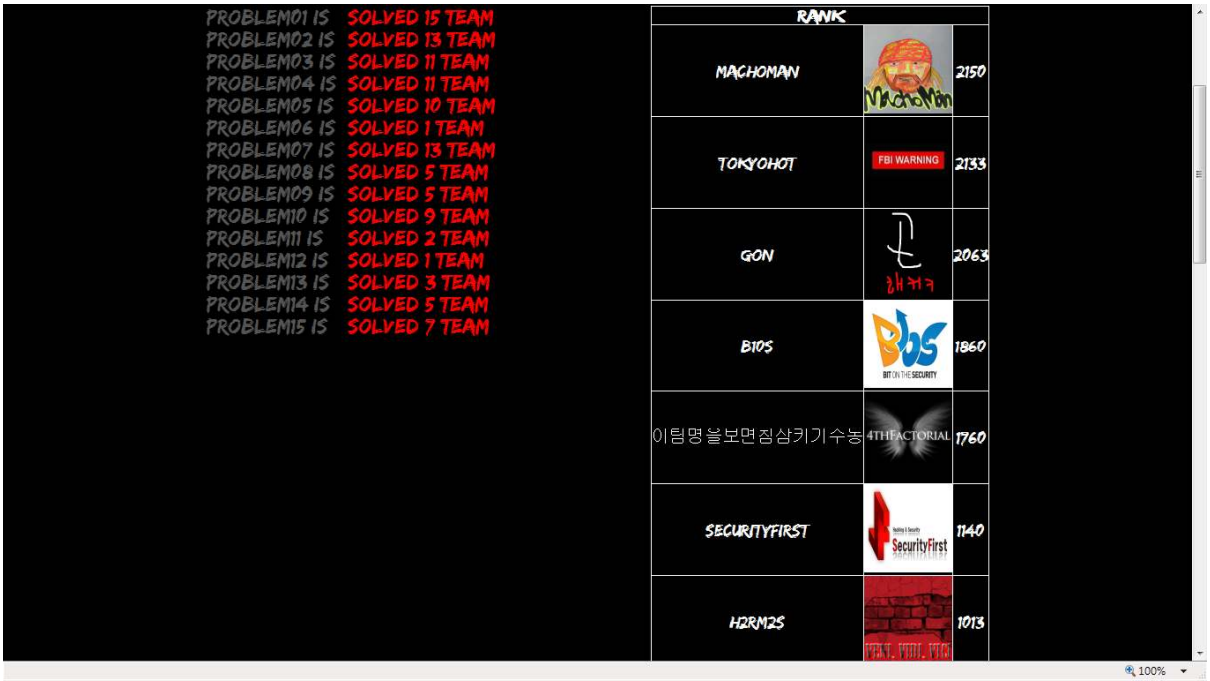
wnsgurzxc@nate.com



목차

0. 대회 결과 & 잡담


CAN YOU SOLVE THIS PROBLEMS?
SCORE 1013 RANK 7
PROBLEM 01 IS REVERSING (100).... SOLVED
PROBLEM 02 IS TRIVIAL (100).... SOLVED
PROBLEM 03 IS SMARTPHONE (100).... SOLVED
PROBLEM 04 IS TRIVIAL (100).... SOLVED
PROBLEM 05 IS SYSTEM (200).... CLICK
PROBLEM 06 IS SECRET (200).... CLICK
PROBLEM 07 IS WEB (200).... SOLVED
PROBLEM 08 IS BINARY (200).... SOLVED
PROBLEM 09 IS WEB (200).... CLICK
PROBLEM 10 IS SECRET (150).... SOLVED
PROBLEM 11 IS CRYPTO (300).... CLICK
PROBLEM 12 IS REVERSING (100).... CLICK
PROBLEM 13 IS REVERSING (53).... SOLVED
PROBLEM 14 IS SYSTEM (300).... CLICK
PROBLEM 15 IS WEB (200).... CLICK


PROBLEM01 IS SOLVED 15 TEAM
PROBLEM02 IS SOLVED 13 TEAM
PROBLEM03 IS SOLVED 11 TEAM
PROBLEM04 IS SOLVED 11 TEAM
PROBLEM05 IS SOLVED 10 TEAM
PROBLEM06 IS SOLVED 1 TEAM
PROBLEM07 IS SOLVED 13 TEAM
PROBLEM08 IS SOLVED 5 TEAM
PROBLEM09 IS SOLVED 5 TEAM
PROBLEM10 IS SOLVED 9 TEAM
PROBLEM11 IS SOLVED 2 TEAM
PROBLEM12 IS SOLVED 1 TEAM
PROBLEM13 IS SOLVED 3 TEAM
PROBLEM14 IS SOLVED 5 TEAM
PROBLEM15 IS SOLVED 7 TEAM
RANK
MACHOMAN 2150
TOKYOHOT 2133
GON 2063
BIOS 1860
이팀명을보면짐삼키기수능 1760
SECURITYFIRST 1140
H2RM2S 1013
100%

이 문서는 ARGOS Hacking Festival 2011 대회를 한 후의 Write-Up이다.
2011-11-28일에 쓰여졌고 대회는 2011-11-26 ~ 2011-11-27 으로 총 2일 동안 진행되었다.
이 Write-Up은 다른 Write-Up들과 다르게 푼 사람이 두 명인 문제도 있다.
주로 문제를 해결한 사람과 도움을 준 사람을 표시하였다.
Ex) Solved By A, B 라면 A가 주로 문제를 풀었고 B가 도움을 준 사람이다.
팀 대회였던 만큼, 팀원들 간에 생각을 공유하며 문제를 풀었다.
그래서 두 명 모두 푼 사람으로 생각, 문서에 기록하였다.
비록 순위도 낮고 푼 문제들도 많진 않지만 우리들의 방식으로 풀었으므로 풀이 과정은 눈여겨볼 만하다고 생각한다.

1. 팀원 소개

Xero – 박준혁 (한국디지털미디어고등학교 1학년)

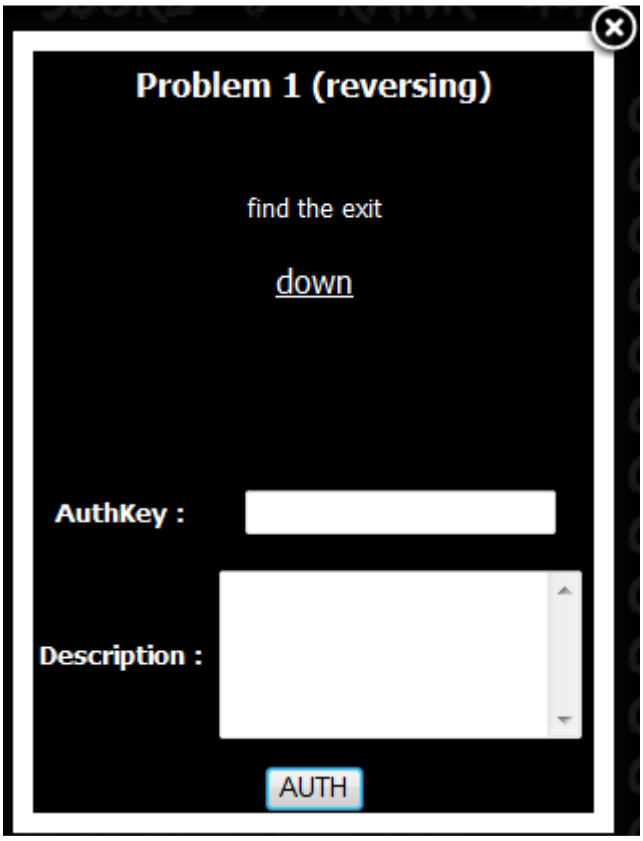
Gogil – 고기완 (한국디지털미디어고등학교 1학년)

Esther – 백한이 (한국디지털미디어고등학교 1학년)

Raymond – 최순형 (한국디지털미디어고등학교 1학년)

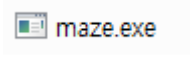
2. 풀이

Problem 01 – Reversing (100) – Solved By Gogil

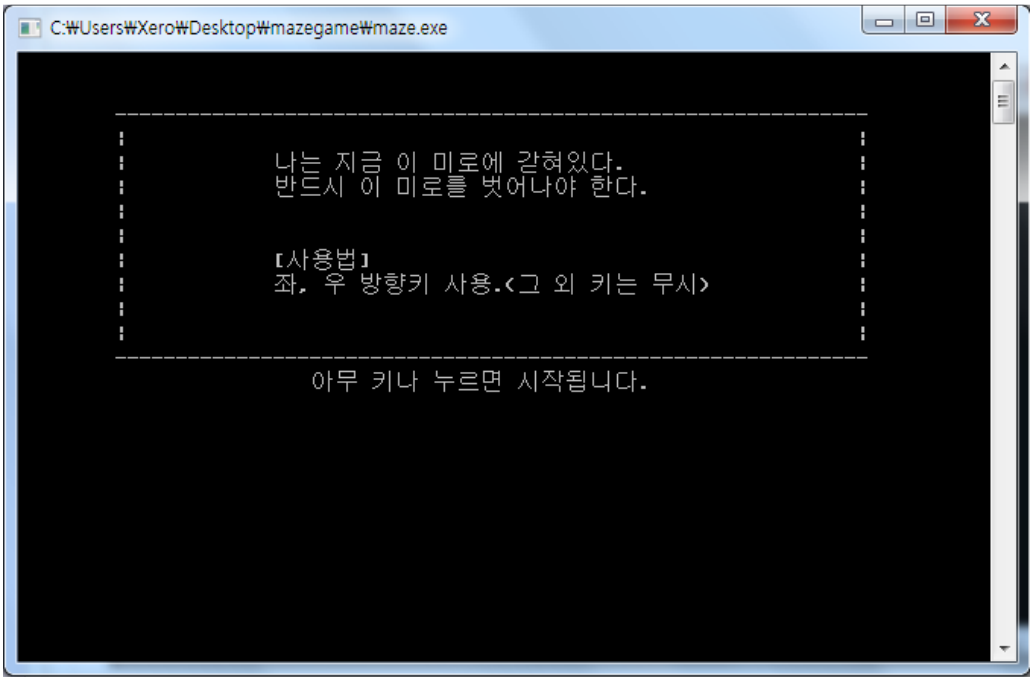


다음과 같이 mazegame.zip 파일 하나를 받게 된다.

압축을 해제하니 maze.exe 파일 하나가 들어있다.

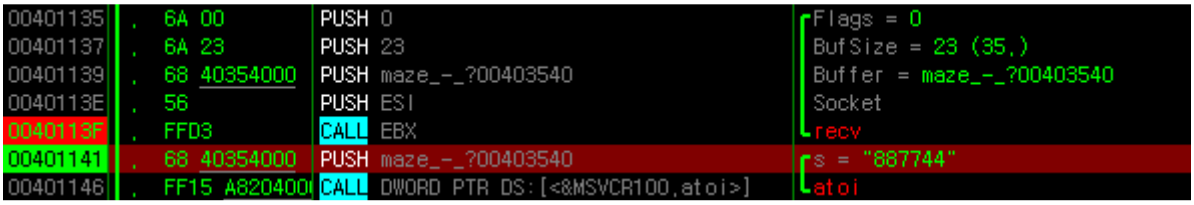


실행시키면 다음과 같이 미로게임이 시작된다.
좌, 우 방향키만을 이용하여 미로에서 벗어나야 한다.



현재 자신의 점수와 왼쪽으로 가게 될 경우 감소되는 점수, 오른쪽으로 가는 경우 감소되는 점수가 나타난다.
그리고 특정한 상황에 따라 go to left, go to right 메세지가 뜨며 해당 방향으로 가지 않으면 게임이 종료된다.

다음과 같이 올리디버거로 열어서 프로그램이 어떻게 돌아가는지 보았다.



0x0040113F에서 서버와 연결 후 목표 점수를 문자열로 받아온다.

그리고 메인 루프에서 왼쪽 점수와 오른쪽 점수를 가져오고 추가로 데이터 하나를 더 받는다.

여기서 추가로 받아오는 데이터가 8일 경우에는 왼쪽으로, 9일 경우에는 오른쪽으로 이동해야 한다.

이를 주의해서 코딩하면 0점이 되었을 때 답이 출력된다.

(단, Total 점수가 수십만 점에서 가끔 20000점 이하로 작게 나오는 경우가 있으므로 20000점 이하일 때에만 작업을 시작하도록 기능을 추가하였다)

다음은 c언어로 작성한 소스이다.

```
#include <stdio.h>
#include <conio.h>
#include <winsock2.h>
#include <windows.h>

#pragma comment(lib, "ws2_32.lib")

int main()
{

    SOCKET      hSock;
    WSADATA     wsaData;
    SOCKADDR_IN servAddr;
    CHAR        cBuf[10];
    INT         nScore, nTotal, nNum1, nNum2, nNum3;

    nTotal = 20001;
    while (nTotal > 20000) {

        if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0)
            return FALSE;

        hSock = socket(PF_INET, SOCK_STREAM, 0);
        if (hSock == INVALID_SOCKET)
            return FALSE;

        memset(&servAddr, 0, sizeof(servAddr));
```

```
    servAddr.sin_family = AF_INET;
    servAddr.sin_addr.s_addr = inet_addr("168.188.130.214");
    servAddr.sin_port = htons(8080);

    if (connect(hSock, (SOCKADDR*)&servAddr, sizeof(servAddr)) == SOCKET_ERROR)
        return FALSE;

    recv(hSock, cBuf, 35, 0);
    nTotal = atoi(cBuf);
    nScore = nTotal;

    if (nTotal > 20000) {
        printf("%d .. ₩n", nTotal);
        closesocket(hSock);
        WSACleanup();
        Sleep(1000);
    }
}

while (1) {
    ZeroMemory(cBuf, 100);
    if (recv(hSock, cBuf, 35, 0) == SOCKET_ERROR) {
        printf("error ₩n");
        return FALSE;
    }
    nNum1 = atoi(cBuf);
    if (nNum1 == 0) {
        printf("Game Over ₩n");
        //return FALSE;
    }

    ZeroMemory(cBuf, 100);
    recv(hSock, cBuf, 35, 0);
    nNum2 = atoi(cBuf);

    ZeroMemory(cBuf, 100);
    recv(hSock, cBuf, 35, 0);
    nNum3 = atoi(cBuf);

    ZeroMemory(cBuf, 100);
    if (nNum3 == 8) {
        itoa(nNum1, cBuf, 10);
        nScore -= nNum1;
    } else if (nNum3 == 9) {
        itoa(nNum2, cBuf, 10);
        nScore -= nNum2;
    } else {
        if ((nScore-nNum1) == 0) {
            itoa(nNum1, cBuf, 10);
            nScore -= nNum1;
        } else if ((nScore-nNum2) == 0) {
```

```
                itoa(nNum2, cBuf, 10);
                nScore -= nNum2;
            } else {
                if (nNum1 > nNum2) {
                    itoa(nNum1, cBuf, 10);
                    nScore -= nNum1;
                } else {
                    itoa(nNum2, cBuf, 10);
                    nScore -= nNum2;
                }
            }
        }
        send(hSock, cBuf, 35, 0);

        printf("(%d %d %d) %d ₩n", nNum1, nNum2, nNum3, nScore);

        if (nScore <= 0) {
            ZeroMemory(cBuf, 100);
            recv(hSock, cBuf, 35, 0);
            printf("%s ₩n₩n", cBuf);
            break;
        }
    }

    closesocket(hSock);
    WSACleanup();
    getch();

    return 0;
}
```

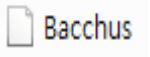
위의 프로그램을 실행시키면 답이 출력된다.

Key : What is the world coming to?

Problem 02 – Trivial (100) – Solved By Gogil

다음과 같이 zip 파일 하나가 주어진다.

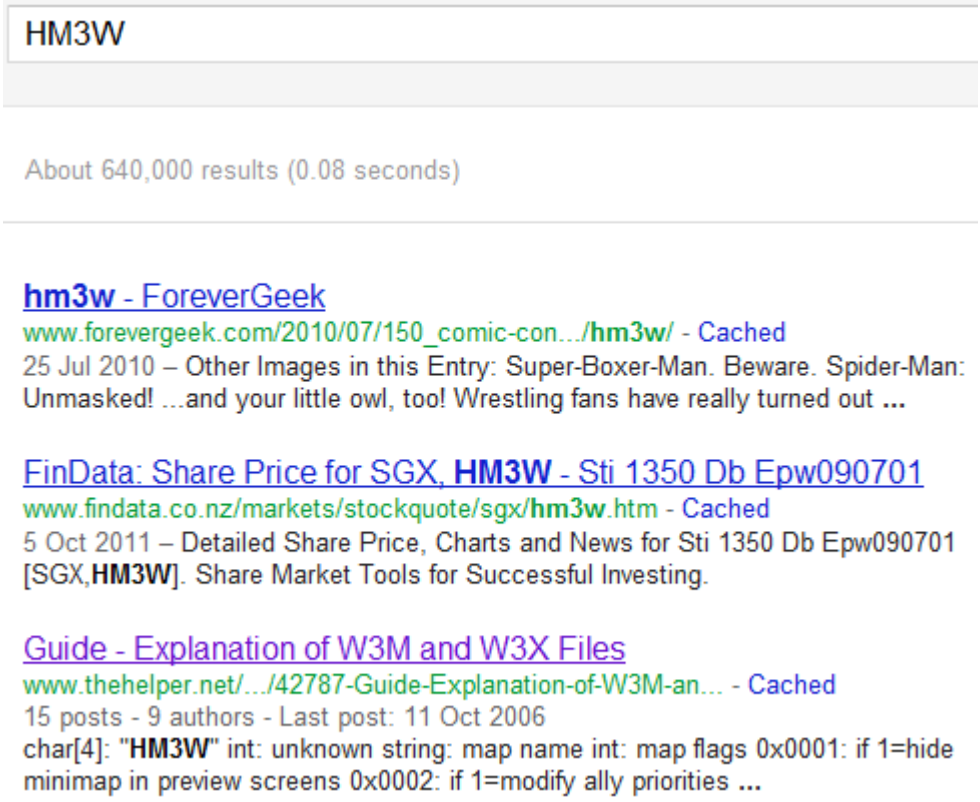
압축을 풀면 Bacchus 파일 하나가 나타난다.



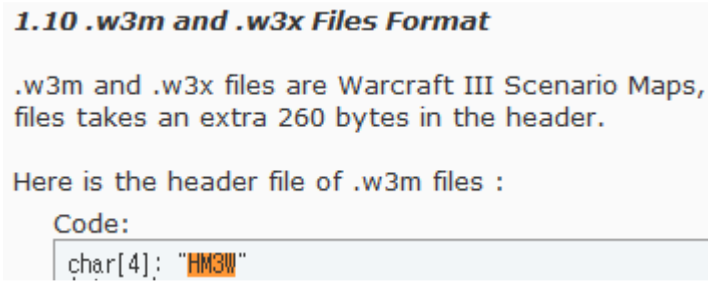
헥스 에디터로 뜯어서 파일 헤더가 HM3W 라는 것을 알아내었다.

```
48 4D 33 57 00 00 00 00 42 61 63 63 68 75 73 00  HM3W....Bacchus.
12 DC 00 00 06 00 00 00 00 00 00 00 00 00 00 00  .Ü..............
```

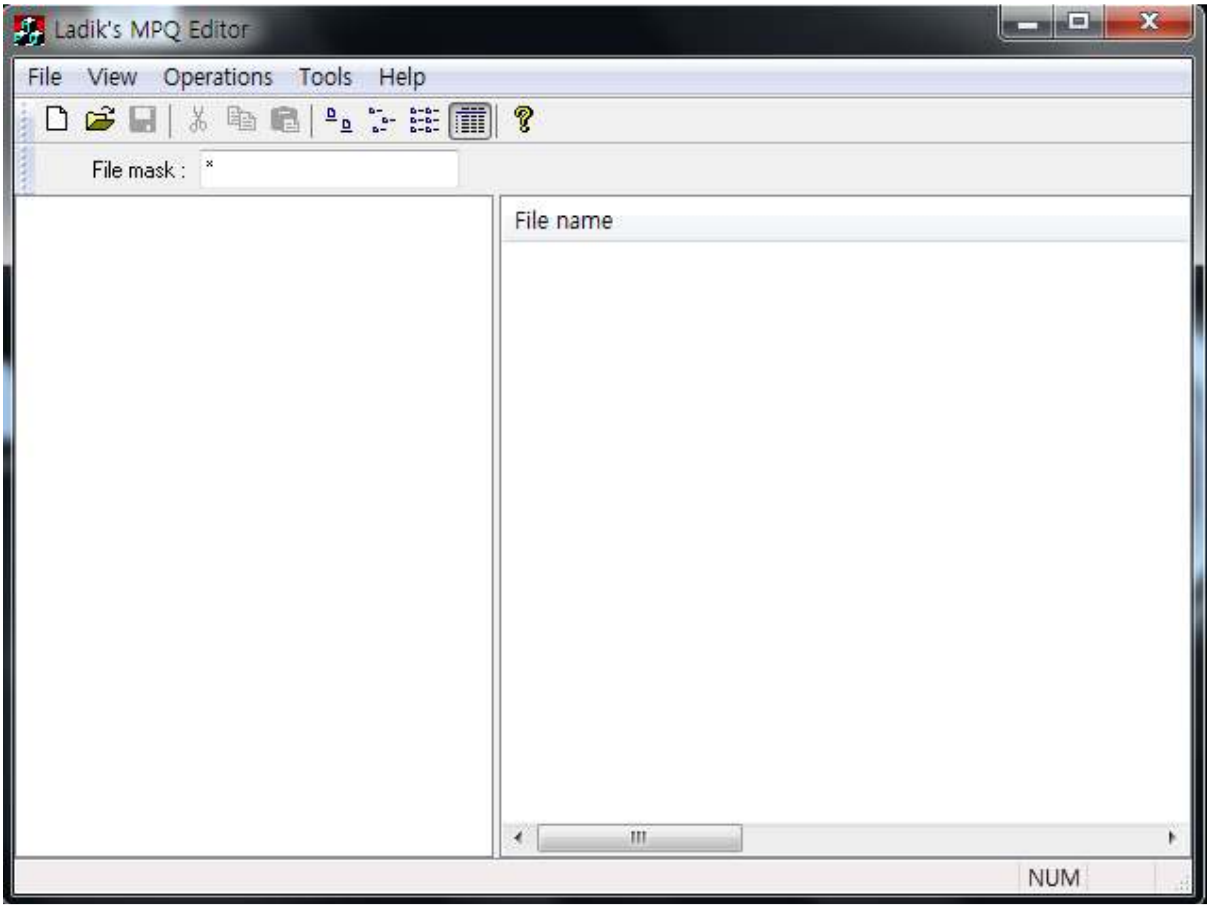
구글에 HM3W를 검색하여 W3M and W3X Files 라는 정보를 얻었다.



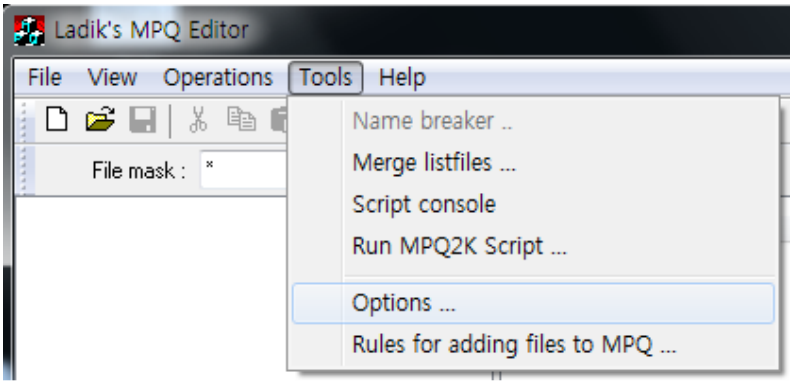
위의 링크를 통해 워크래프트3 맵 파일이라는 것을 알게 되었다.



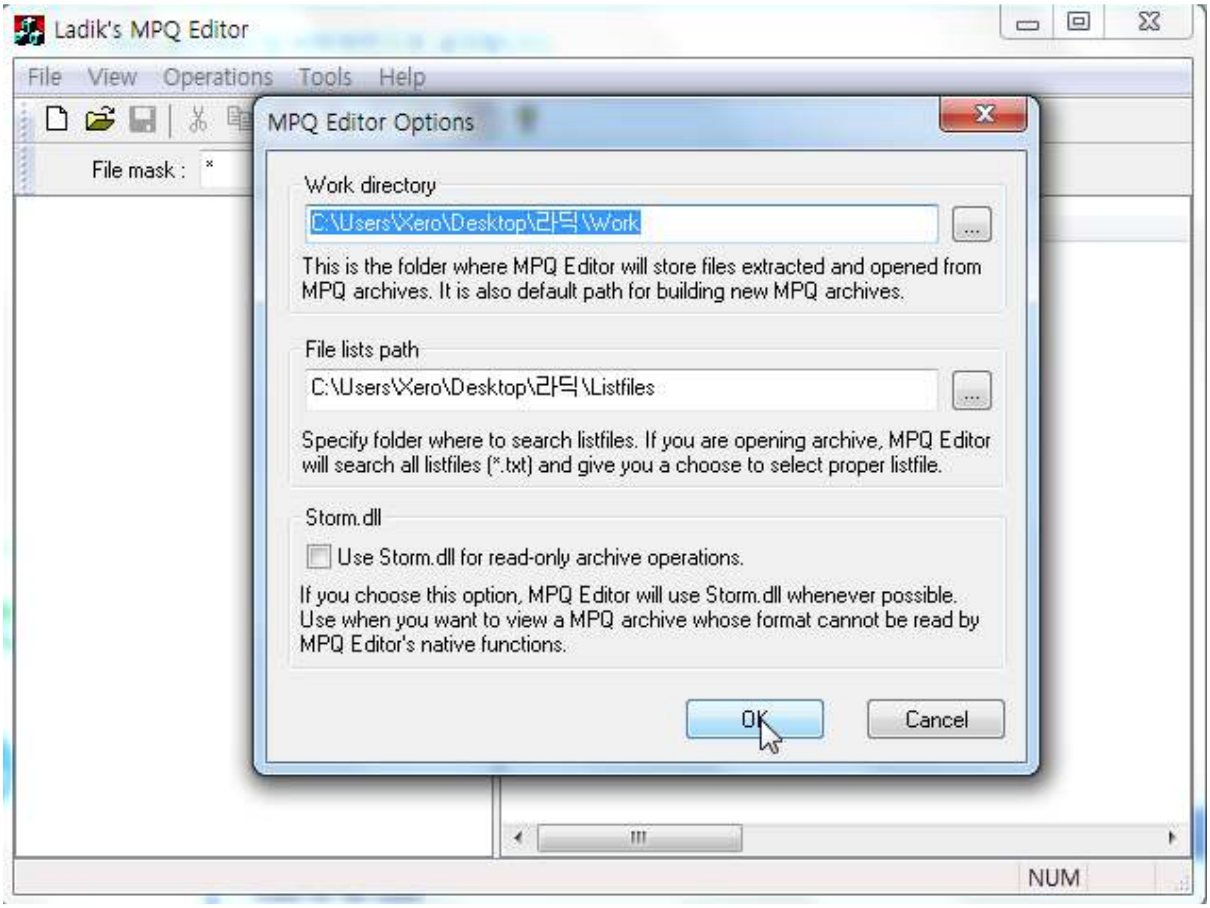
워크래프트3 맵 에디터를 검색하다가 라딕이라는 프로그램을 발견하였다.



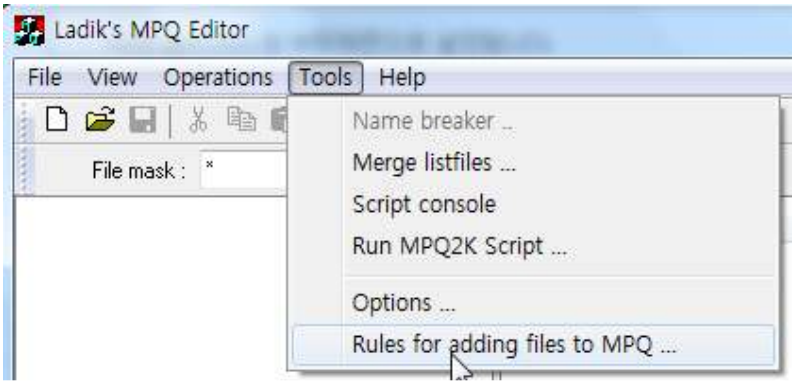
라딕을 사용하려면 우선 설정을 해야 하기에 Tools의 Options에 들어갔다.



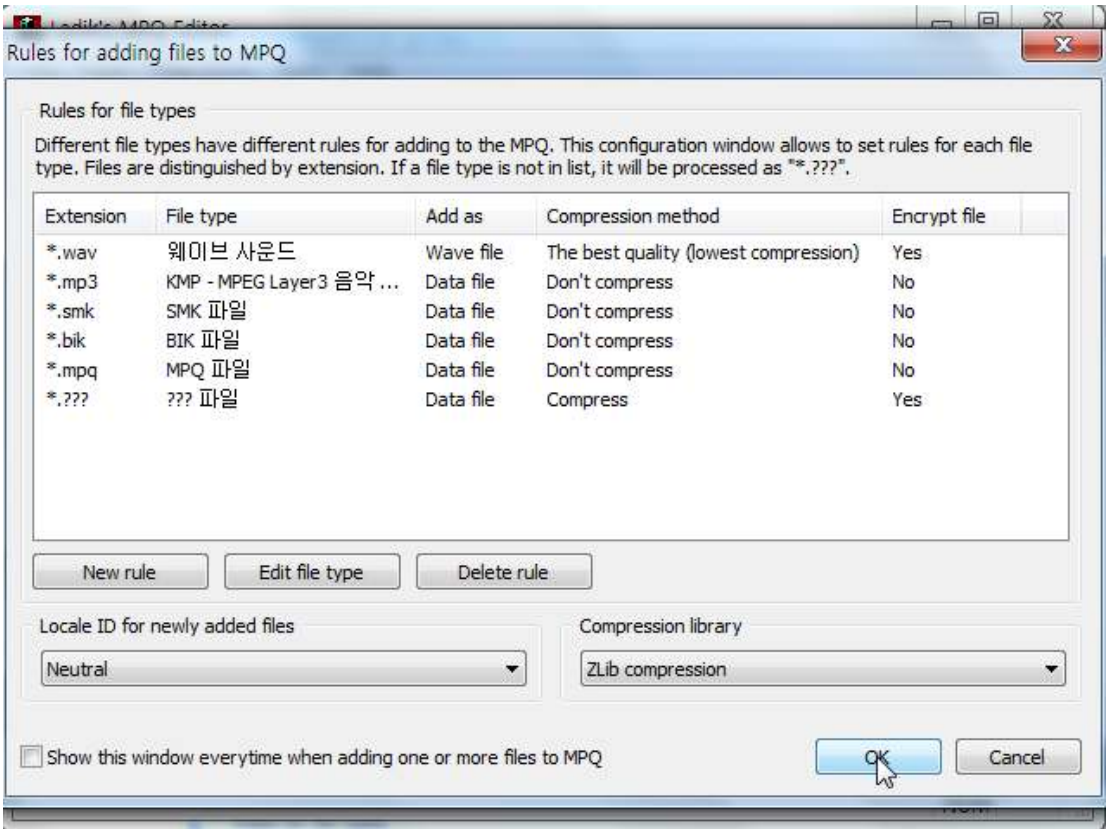
다음과 같이 우선 경로를 설정하였다.



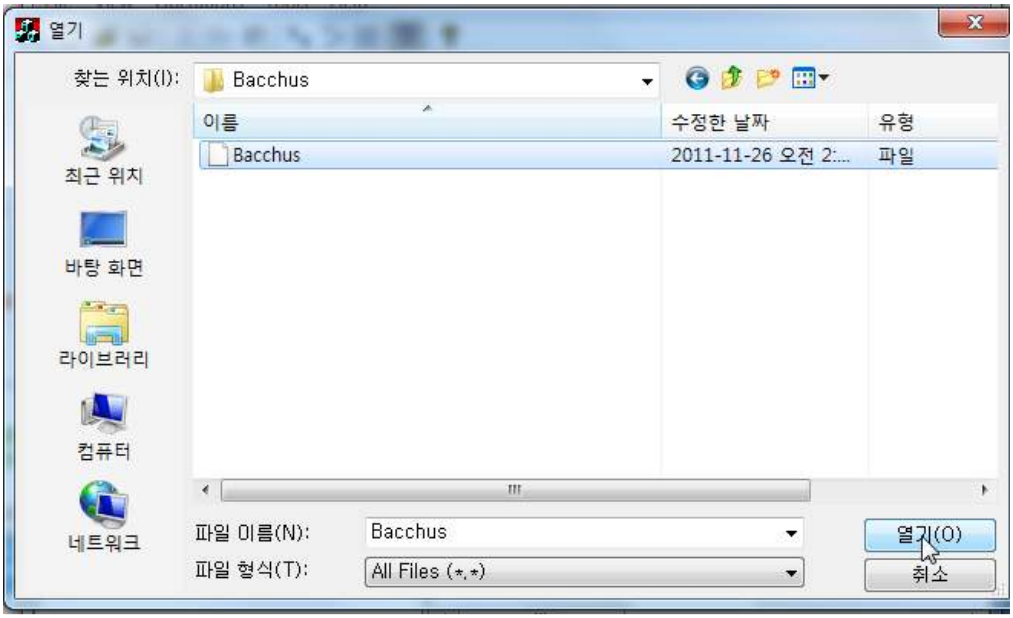
그리고 Tools에 Rules for adding files to MPQ로 들어갔다.



그러면 다음과 같은 창이 뜨는데, Compression library를 ZLib compression으로 변경하고 Show this window everytime when adding one or more files to MPQ 의 체크를 풀면 된다.

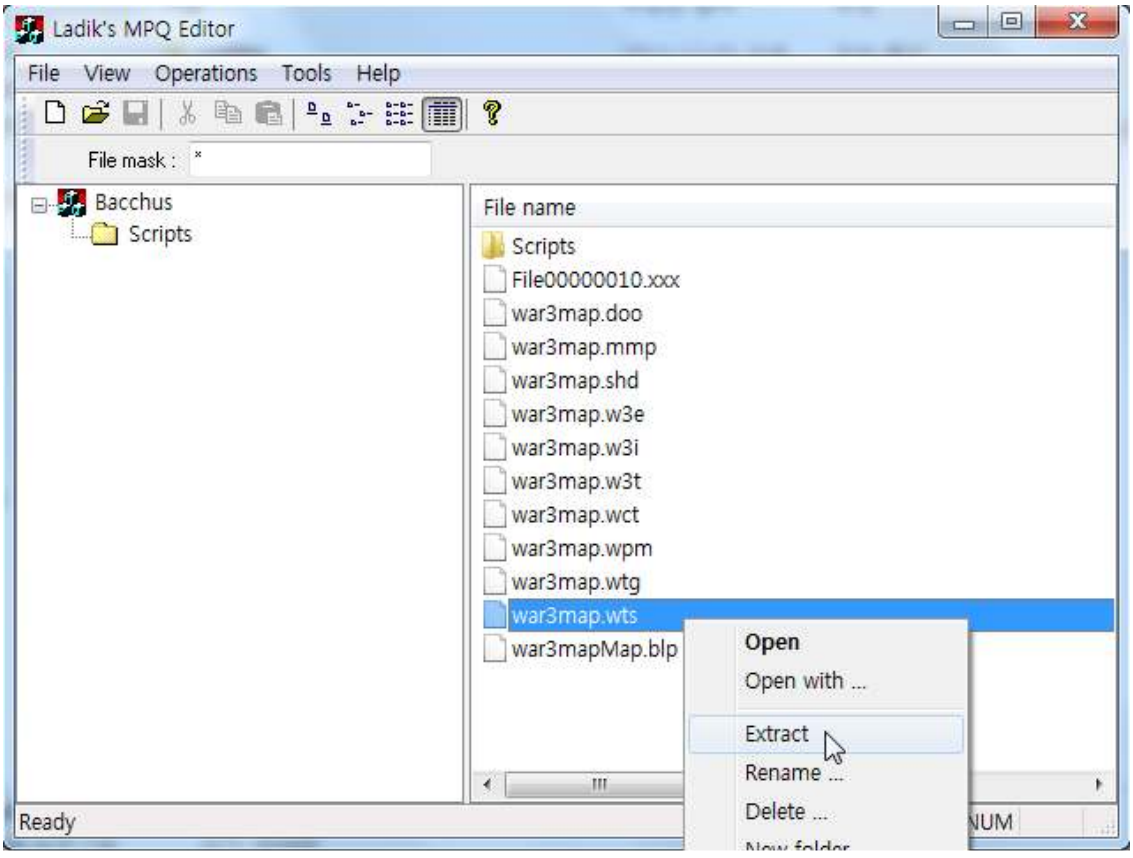


그리고 파일 형식을 All Files로 변경한 후 문제 파일을 불러들였다.



그러면 다음과 같은 창이 하나 뜨는데, Warcraft Ⅲ.txt 를 선택하고 OK 버튼을 누르면 된다.

그러면 다음과 같이 여러 파일들이 뜨게 되는데 war3map.wts 파일을 Extract 하였다.



그리고 메모장으로 열어보니 다음과 같이 base85들이 보였다.

STRING 12
// 아이템: kygh (고스트 키), Description (설명)
{
<~6#I~>
}

STRING 13
// 아이템: kygh (고스트 키), Ubertip (도구 도움말 - 확장)
{
<~6#I~>
}

STRING 14
// 아이템: mgtk (마법 키 체인), Description (설명)
{
꽝
}

STRING 15
// 아이템: mgtk (마법 키 체인), Ubertip (도구 도움말 - 확장)
{
꽝
}

STRING 16
// 아이템: kymn (문 키), Description (설명)
{
<~B5_iuC^g^~>
}

STRING 17

// 아이템: kymn (문 키), Ubertip (도구 도움말 - 확장)
{
<~B5_iuC^g^~>
}

STRING 18
// 아이템: kybl (블러드 키), Description (설명)
{
<~Bkq9eG@:~>
}

STRING 19
// 아이템: kybl (블러드 키), Ubertip (도구 도움말 - 확장)
{
<~Bkq9eG@:~>
}

STRING 20
// 아이템: kysn (태양의 열쇠), Ubertip (도구 도움말 - 확장)
{
<~Eas,uAooX~>
}

STRING 21
// 아이템: kysn (태양의 열쇠), Description (설명)
{
<~Eas,uAooX~>
}

순서대로 base85만 나열하면 다음과 같이 된다.

<~6#I~>

<~B5_iuC^g^~>

<~Bkq9eG@:~>

<~Eas,uAooX~>

디코더에 넣고 돌리니 다음의 값이 나왔다.

Ar

gos_l

ike_wa

rcraft.

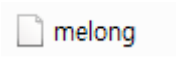
위의 문자열들을 붙여서 인증하니 문제가 클리어되었다.

Key : Argos_like_warcraft.

Problem 03 – Smart Phone (100) – Solved By Xero, Gogil

다운로드를 누르면 다음과 같이 zip 파일 하나를 받을 수 있다.

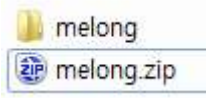
압축을 해제하자 다음과 같이 확장자가 없는 파일 하나가 나왔다.



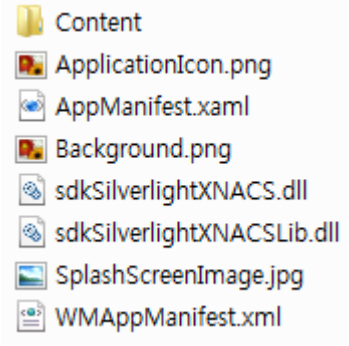
헥스 에디터로 파일 시그너처를 보니 PK 로 압축 파일인 것을 알 수 있었다.

```
50 4B 03 04 14 00 00 08 08 00 56 71 2A 3F 08 E4  PK........Vq*?.ä
71 08 D2 05 00 00 CD 05 00 00 13 00 00 00 41 70  q.Ò...Í.......Ap
70 6C 69 63 61 74 69 6F 6E 49 63 6F 6E 2E 70 6E  plicationIcon.pn
```

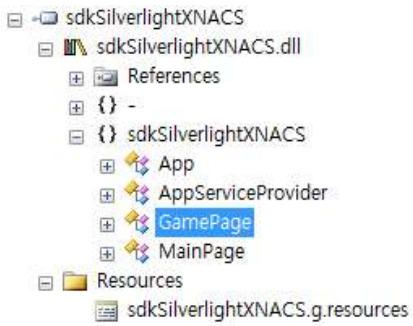
다음과 같이 확장자를 zip으로 바꾸고 압축을 해제하였다.



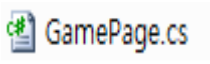
그러자 다음과 같은 파일들이 나왔다.



Reflector로 뜯어보니 GamePage 소스 부분이 메인 소스라는 생각이 들었다.



다음과 같이 GamePage 소스만을 뽑아보았다.



소스를 보던 중 다음과 같이 소스가 있었다.

```
private string key = "JE_KiSS_Chocolate";
```

위의 소스를 보고 답이 JE_KiSS_Chocolate 라고 생각했었으나 정답이 아니었다.
소스를 다시 살펴보자 수상한 소스들이 발견되었다.

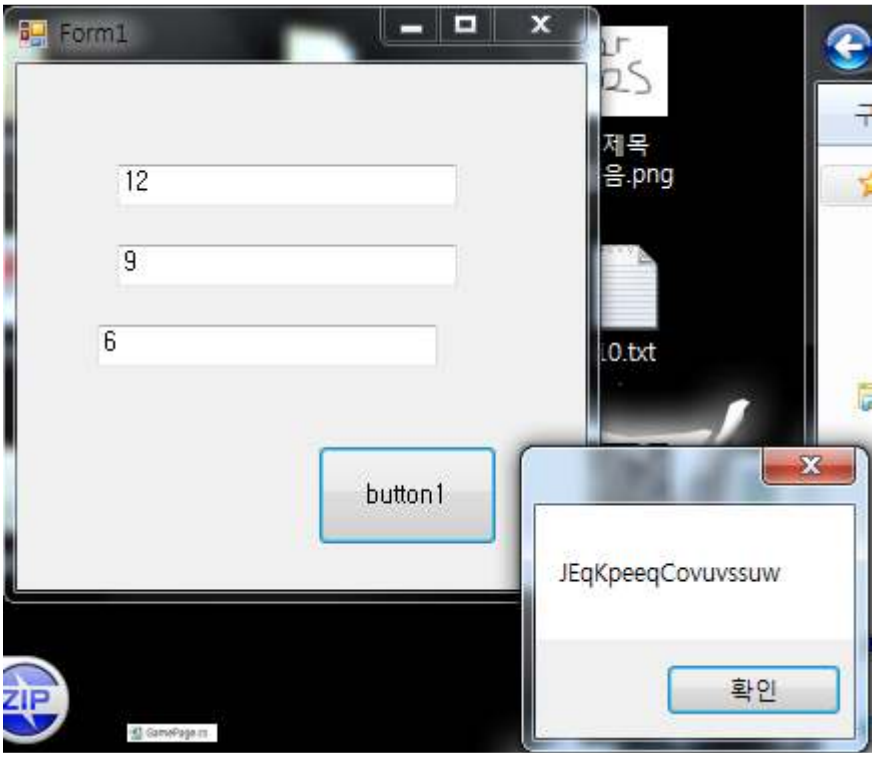
```
private char[] table = new char[] {
    'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P',
    'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f',
    'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
    'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '+', '/',
    'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P',
    'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f',
    'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
    'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '+', '/',
    'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P',
    'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f',
    'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
    'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '+', '/'
};
```

```
private int getResult()
{
    return (((((5 * this.x) * this.x) * this.x) - (((7 * this.y) * this.y) * this.y)) + ((((2 * this.z) * this.z) * this.z) * this.z));
}
```

```
private void showmethemoney()
{
    char[] chars = new char[this.key.Length + 1];
    chars = Encoding.UTF8.GetChars(Encoding.UTF8.GetBytes(this.key));
    for (int i = 0; i < this.key.Length; i++)
    {
        chars[i] = this.table[chars[i] + this.x];
        chars[i] = this.table[chars[i] - this.y];
        chars[i] = this.table[chars[i] - this.z];
    }
    string str = new string(chars);
    if (this.getResult() == 0x17f1)
    {
        MessageBox.Show(str, "Is this a key?", 0);
    }
}
```

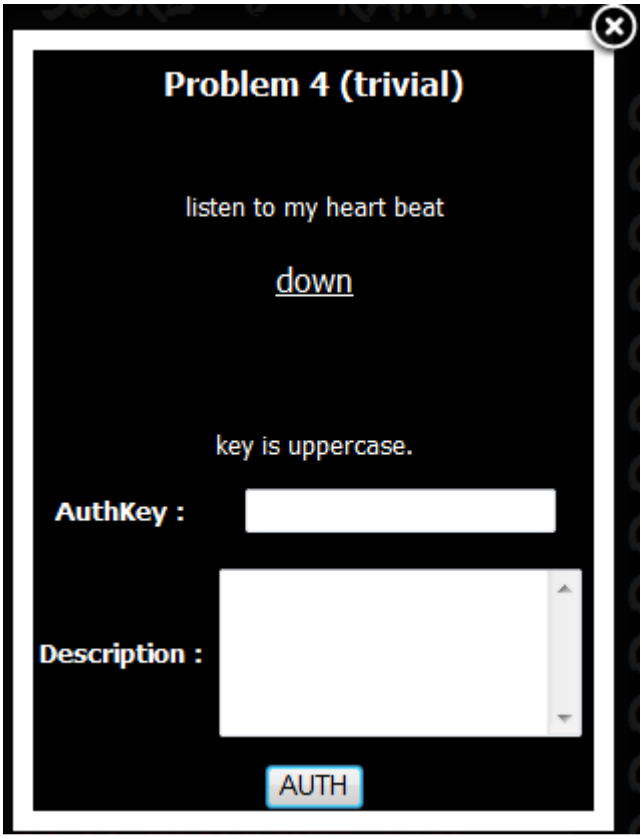
getResult() 에서 얻은 x, y, z 값을 이용해 테이블에서 문자를 뽑아내 답을 만든다는 것을 알 수 있었다.
getResult() 부분을 for문으로 브루트포싱하여 x=12, y=9, z=6 이라는 값을 얻었다.
위의 소스들을 토대로 x, y, z 값을 입력받아 위의 처리 과정을 거쳐 테이블에서 문자를 뽑아내는 프로그램을 만들었다.

x에 12, y에 9, z에 6의 값들을 넣고 프로그램을 실행시키자 JEqKpeeqCovuvssuw 라는 값이 나왔고 인증에 성공했다.



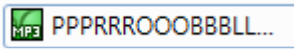
Key : JEqKpeeqCovuvssuw

Problem 04 – Trivial (100) – Solved By Esther, Xero



다음과 같이 zip 파일 하나가 주어진다.

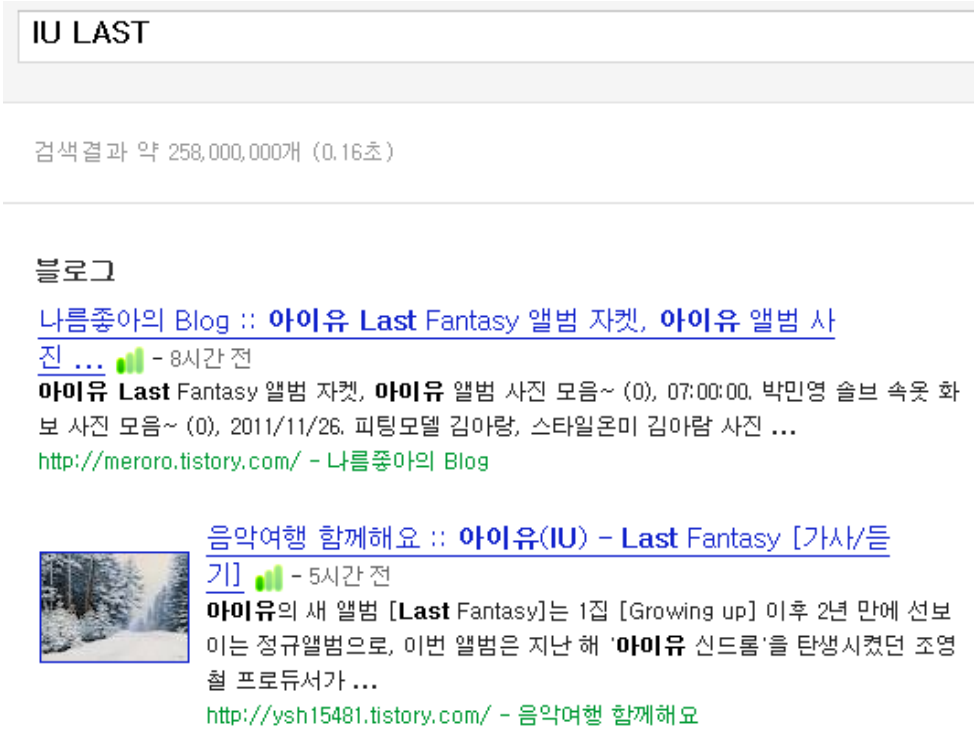
압축을 풀자 다음과 같은 mp3 파일이 있었다.



다음과 같이 정상적으로 재생이 되었고 휴대폰의 기본 버튼음 소리로 추정되는 소리들이 들렸다.

주어진 음원이 휴대폰의 기본 버튼음 소리라고 예상하였다.
누른 번호에 맞는 영어들을 모으면 키가 될 것이라고 생각했다.
그냥 들어서 소리를 알아 맞추기에는 음도 정확히 모르거니와 빠르기도 너무 빨라서 힘들다고 생각했다.
그래서 Mixcraft 라는 프로그램을 사용하여 다음과 같이 음원을 1/4배속으로 늦춘 후 버튼 음 횟수 별로 구간을 나누어서 듣기 좋게 하였다.

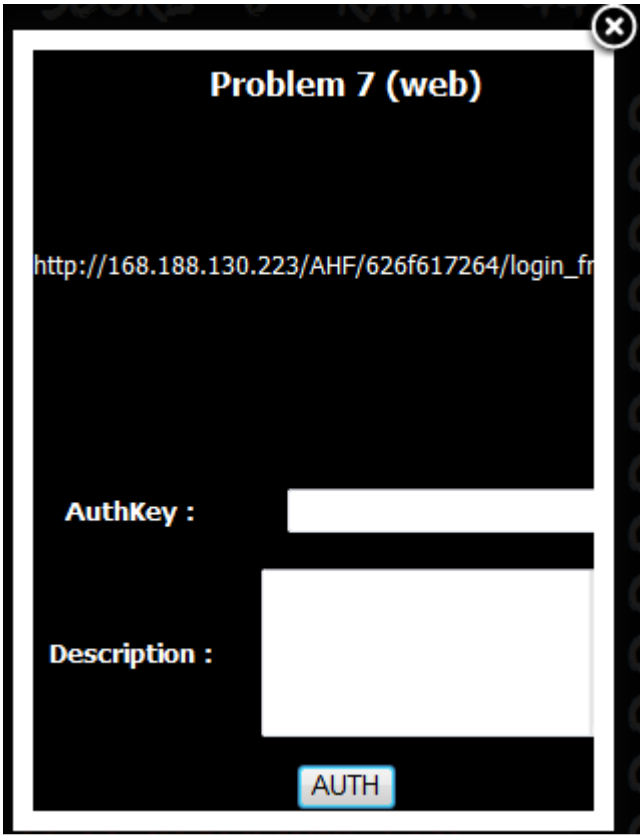
처음에는 부모님 세대의 2G 폰의 소리라고 생각하였다.
1, 2, 3, 4, 5 가 각각 도, 레, 미, 파, 솔이 되어서 비행기 같은 간단한 동요도 휴대폰 자판으로 연주 할 수 있듯이 말이다.
그러나 2G 폰들은 3G 폰으로 바꾼 지 오래되어 집이나 주변 사람들에게서도 2G 폰을 구할 수 없었다.
인터넷에 검색해 봐도 없어서 대충 끼워 맞추며 풀려던 참에 팀원이 휴대폰 통화 중에 누르는 버튼음 소리가 아니냐고 했다.
기억을 더듬어 생각해 보니 소리가 정확히 일치했고, 직접 통화 중에 버튼을 눌러보니 같은 소리가 났다.
일일히 통화 중에 버튼을 누르며 소리를 비교하기에는 무리가 있다고 생각하여 1, 2, 3, 4, 5, 6, 7, 8, 9, *, 0, # 을 한 번씩 눌러서 그 소리들을 녹음하였다.
절대음감이 아니라 다 비슷비슷하여 구분이 안되어서 음악을 전공하였던 팀원의 도움을 받아 소리를 비교하면서 버튼들을 알아내었다.
44488555277778 까지 확실히 알아냈다고 생각하여 영어로 바꾸니 IULAST 라는 글자가 완성되었다.
구글링으로 IU LAST 를 검색하니 IU가 LAST FANTASY 라는 새 앨범을 냈다는 사실을 알게 되었고 IULASTFANTASY로 인증에 성공했다.



다음은 음원에서 눌렀던 버튼들이다.
444885552777783332668277779999

Key : IULASTFANTASY

Problem 07 - Web (200) – Solved By Xero



회원가입을 하고 로그인하여 보면 게시판이 있다.
게시물을 읽을 때 no라는 파라미터에서 SQL Injection공격이 가능하다는 것을 알아내었다.
Information.schema에서 테이블과 컬럼들을 뽑아보면 Golden_Key라는 테이블 안에 key1, key2, key3, key4, key5, key6 이라는 값들이 존재하는 것을 볼 수 있다.

다음과 같이 SQL Injection을 시도하여 key들을 빼왔다.
http://168.188.130.223/AHF/626f617264/read.php?no=5%20and%209=0%20union%20select%202,3,(select%20key1%20from%20Golden_Key),1%23

다음은 key1~6이다.
whiterick, isthe, best, rich, hacker, inKorea

key들을 순서대로 나열하여 인증하니 답이었다.

Key : whiterickisthebestrichhackerinKorea

Problem 08 – Binary (200) – Solved By Gogil

다음과 같이 zip 파일 하나가 주어진다.

eggbread.zip

압축을 풀면 다음과 같이 egg라는 파일 하나가 나온다.

egg

헥스 에디터로 열어서 헤더를 보니 .ELF 였다.

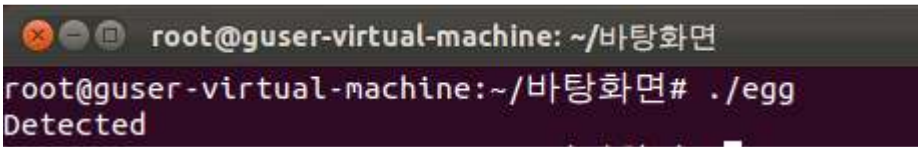
```
7F 45 4C 46 01 01 01 00 00 00 00 00 00 00 00 00  .ELF............
02 00 03 00 01 00 00 00 28 A0 04 08 34 00 00 00  ........( ..4...
```

검색을 통해 리눅스상의 실행파일이라는 것을 알아내었다.

```
7F 45 4C 46                                      .ELF
```

n/a Executable and Linking Format executable file (Linux/Unix)

리눅스로 파일을 옮겨서 실행해보니 다음과 같이 Detected라는 문구가 출력되고 프로그램이 종료되었다.



디스어셈블하여 프로그램 진입점부터 차례로 살펴보면 다음과 같이 한 바이트씩 메모리를 채우는 부분이 존재한다.
이러한 부분이 굉장히 많은데, 윈도우 바이너리의 언패킹 루틴과 흡사한 것을 알 수 있다.
저렇게 수 많은 쓰기 작업이 끝나고 Detected 라는 메시지를 띄울 것인지 확인하는 분기점이 존재한다.

```
0804:a091 85 c9                 test ecx, ecx
0804:a093 74 05                 jz 0x0804a09a
0804:a095 88 02                 mov byte ptr [edx], al
0804:a097 83 c2 01              add edx, 1
0804:a09a c6 85 e3 fc ff ff 63  mov byte ptr [ebp-797], 99
0804:a0a1 c6 85 e4 fc ff ff 7c  mov byte ptr [ebp-796], 124
0804:a0a8 c6 85 e5 fc ff ff 77  mov byte ptr [ebp-795], 119
0804:a0af c6 85 e6 fc ff ff 7b  mov byte ptr [ebp-794], 123
0804:a0b6 c6 85 e7 fc ff ff f2  mov byte ptr [ebp-793], 0xf2
0804:a0bd c6 85 e8 fc ff ff 6b  mov byte ptr [ebp-792], 107
0804:a0c4 c6 85 e9 fc ff ff 6f  mov byte ptr [ebp-791], 111
0804:a0cb c6 85 ea fc ff ff c5  mov byte ptr [ebp-790], 0xc5
0804:a0d2 c6 85 eb fc ff ff 30  mov byte ptr [ebp-789], 48
0804:a0d9 c6 85 ec fc ff ff 01  mov byte ptr [ebp-788], 1
0804:a0e0 c6 85 ed fc ff ff 67  mov byte ptr [ebp-787], 103
```

다음과 같이 0x0804E3A7과 0x0804E40C부분을 점프하도록 수정하였다.

```
0804:e3a0 83 bd 40 f4 ff ff 00  cmp dword ptr [ebp-3008], 0
0804:e3a7 79 22                 jns 0x0804e3cb
0804:e3a9 8d 45 c3              lea eax, [ebp-61]
0804:e3ac 50                    push eax
0804:e3ad b8 04 00 00 00        mov eax, 4
```

```
0804:e406 3b 85 38 f4 ff ff     cmp eax, dword ptr [ebp-3016]
0804:e40c 74 33                 jz 0x0804e441
0804:e40e 8b 85 34 f4 ff ff     mov eax, dword ptr [ebp-3020]
0804:e414 83 c0 01              add eax, 1
0804:e417 3b 85 3c f4 ff ff     cmp eax, dword ptr [ebp-3012]
```

그리고 실행하면 Detected 메시지는 뜨지 않지만 0x08048E0에서 메모리 참조 오류가 발생한다.
오류가 발생하는 부분을 NOP로 처리하고 실행하면 프로그램은 다음과 같이 문자열을 입력받는다.

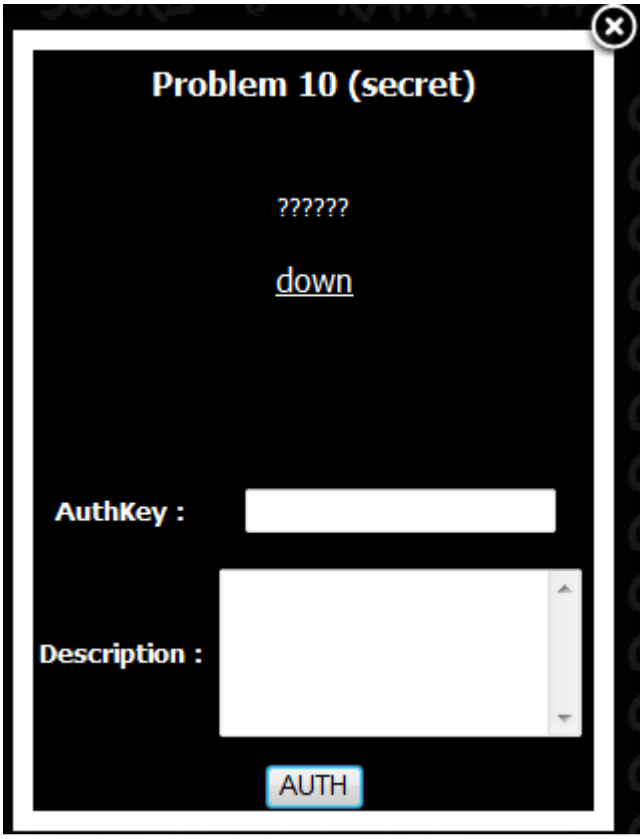
input데이터를 입력하고 wrong input 메시지가 뜬 후, 메모리에서 문자열을 검색해보면 chiffoncake 이라는 문자열을 발견할 수 있다.

```
08048560: UWVS
080485bc: [^_]
08048630: input :
0804863c: chiffoncake
08048648: ALYYHNYUWBCZZIHWUEY
0804865c: wrong input
080486d3: ;*2$\"$
```

Input에 chiffoncake을 입력하면 위에 보이는 ALYYHNYUWBCZZIHWUEY 라는 문자열이 나온다.
출력된 ALYYHNYUWBCZZIHWUEY 를 ROT-6 시키면 답이 나온다.

Key : GREENTEACHIFFONCAKE

Problem 10 – Secret (150) – Solved By Gogil, Xero



다음과 같이 zip 파일 하나가 주어진다.

oroioroiclnt.zip

압축을 풀면 clnt 라는 파일 하나가 존재한다.

clnt

헥스 에디터로 열어서 파일 헤더를 보니 .ELF 로, 위의 문제와 같이 리눅스상의 실행파일이라는 것을 알게 되었다.

```
7F 45 4C 46 01 01 01 00 00 00 00 00 00 00 00 00  .ELF............
02 00 03 00 01 00 00 00 F0 85 04 08 34 00 00 00  ........ð…..4...
```

다음은 IDA로 디스어셈블해 본 주요 소스이다.

```
v14 = *MK_FP(__GS__, 20);
memset(&v10, 0, 0x400u);
memset(&v9, 0, 0x400u);
memset(&v8, 0, 0x400u);
v5 = 0;
v4 = socket(2, 1, 0);
if ( v4 >= 0 )
{
  memset(&v11, 0, 0x10u);
  *(_WORD *)&v11 = 2;
  v13 = inet_addr("168.188.130.214");
  v12 = htons(0x35u);
  if ( connect(v4, (const struct sockaddr *)&v11, 0x10u) == -1 )
  {
    result = 1;
  }
  else
  {
    memset(&v10, 0, 0x400u);
    read(v4, &v10, 0x400u);
    puts(&v10);
    while ( 1 )
    {
      v3 = v5++ <= 31;
      if ( !v3 )
        break;
      printf("> ");
      fgets(&v10, 1024, stdin);
      memset(&v8, 0, 0x400u);
      sprintf(&v8, "oroi@%s", &v10);
      v1 = strlen(&v8);
      if ( write(v4, &v8, v1) <= 0 )
        break;
      memset(&v10, 0, 0x400u);
      if ( read(v4, &v10, 0x400u) <= 0 )
        break;
      if ( strlen(&v10) > 1 )
      {
        v7 = 1;
        v6 = 0;
        memset(&v9, 0, 0x400u);
        while ( *(&v10 + v7) != 124 )
          *(&v9 + v6++) = *(&v10 + v7++);
        v2 = strchr(&v10, 124);
        printf("--> %s¥n", v2 + 1);
      }
    }
    close(v4);
    result = 0;
  }
}
else
{
  result = 1;
}
if ( *MK_FP(__GS__, 20) != v14 )
  __stack_chk_fail();
return result;
```

이 바이너리는 해당 서버과 포트로 접근하여 입력을 하면 데이터를 받아오는 형식이다.

그러나 | 부분을 파싱하여 답들을 다 잘라버린다.

그래서 다음과 같이 코딩을 해서 문자열들을 잘리지 않게 모두 받아보았다.

```
#include <stdio.h>
#include <string.h>
#include <winsock2.h>
#include <windows.h>


#pragma comment(lib, "ws2_32.lib")



int main()
{
	SOCKET		hSock;
	WSADATA		wsaData;
	SOCKADDR_IN servAddr;
	CHAR		cBuf[0x10000];


	if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0)
		return FALSE;


	hSock = socket(PF_INET, SOCK_STREAM, 0);
	if (hSock == INVALID_SOCKET)
		return FALSE;



	memset(&servAddr, 0, sizeof(servAddr));
	servAddr.sin_family = AF_INET;
	servAddr.sin_addr.s_addr = inet_addr("168.188.130.214");
	servAddr.sin_port = htons(53);

	if (connect(hSock, (SOCKADDR*)&servAddr, sizeof(servAddr)) == SOCKET_ERROR)
		return FALSE;


	ZeroMemory(cBuf, 0x10000);
	recv(hSock, cBuf, 0x10000, 0);
	printf("%s ₩n", cBuf);

	while (1) {
		strcpy(cBuf, "oroi@");
		memset(&cBuf[5], 'a', 0x10000);
```

```
                //sprintf(cBuf, "oroi@%s", cBuf);
                //sprintf(cBuf, "oroi@What are you doing????");
                send(hSock, cBuf, 0x200, 0);

                ZeroMemory(cBuf, 0x10000);
                if (recv(hSock, cBuf, 0x10000, 0) == SOCKET_ERROR) break;
                printf("%s ₩n", cBuf);
        }



        closesocket(hSock);
        WSACleanup();

        return 0;
}
```

받은 문자열들은 다음과 같다.

Welcome to argos 2011 CTF !!!!

enjoy your challenge

M23|What are you doing????

H23|Do you want answer????

I26|But This is not answer!!!

w25|Don't try Brute Force!!!

a10|WaiT!!!!!

S20|beep, It's Fake!!!!

B24|It's not BOF problem!!!

p30|Try something other method!!!

c20|Hack the planet!!!!

y26|You can try everything!!!

B27|But don't attack server!!!

h26|We don't have big money!!

M38|so..... our server is normal PC...T.T

X26|What the fuck@!!!!@!@!@!@

d32|This x-mas will be lonely!!!T.T

h45|Why we have to spend my time with This PC!!!

e35|Argos Threat me... make ploblem!!!

X27|Exso merong!!!! Yak o-r-ji

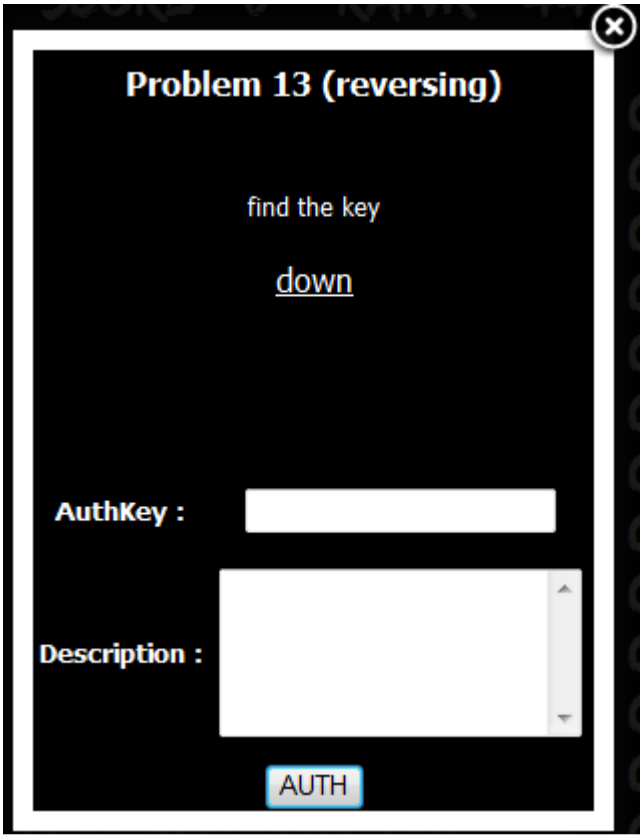
M24|Do you know desert fox?
g34|Whiterick is famous rich in korea
M19|Blues were defeat!
T14|what the hell
B67|knock!!knock!!penny~!!knock!!knock!!penny~!!knock!!knock!!penny~!!
u17|Suck my asshole!
Z22|ma boy~ ma boy~ baby~
T31|Girls Generation's The boys!!!
F24|Secret's love move~~!!!
517|We like K-POP!!!
I21|gam sa hap ni da!!!!
S22|in english, thanks!!!
E22|in china, chez~ chez!
h42|This is last chance to catch your word!!!

| 앞쪽의 문자들만 뽑아내 이어서 다음과 같은 문장을 만들어냈다.
MHIwaSBpcyBhMXdheXMgMTBuZTF5ISEh

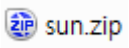
Base64 디코드를 하니 다음의 문장이 나왔고 인증에 성공했다.
0r0i is a1ways 10ne1y!!!

Key : 0r0i is a1ways 10ne1y!!!

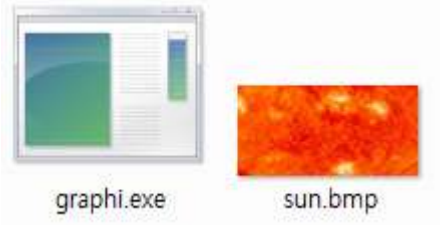
Problem 13 – Reversing (53) – Solved By Gogil



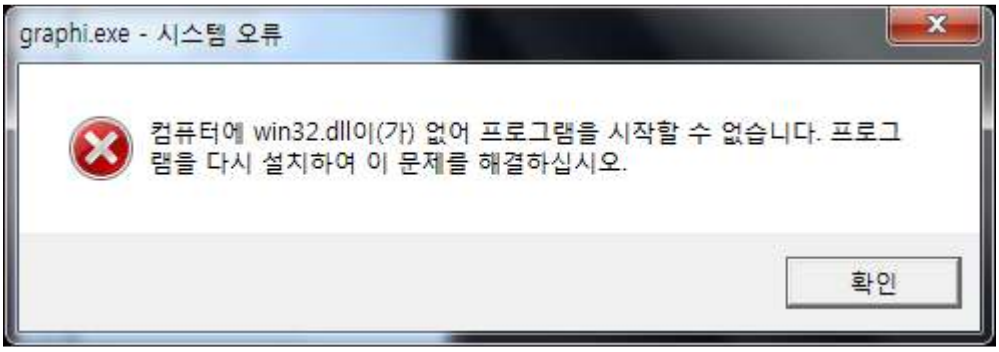
sun.zip 이라는 압축파일 하나가 문제로 주어진다.



압축을 풀면 다음과 같이 graphi.exe 파일과 sun.bmp 파일이 나온다.



graphi.exe 파일을 실행시키자 다음과 같이 win32.dll이 없다며 에러를 출력하고 프로그램이 종료된다.

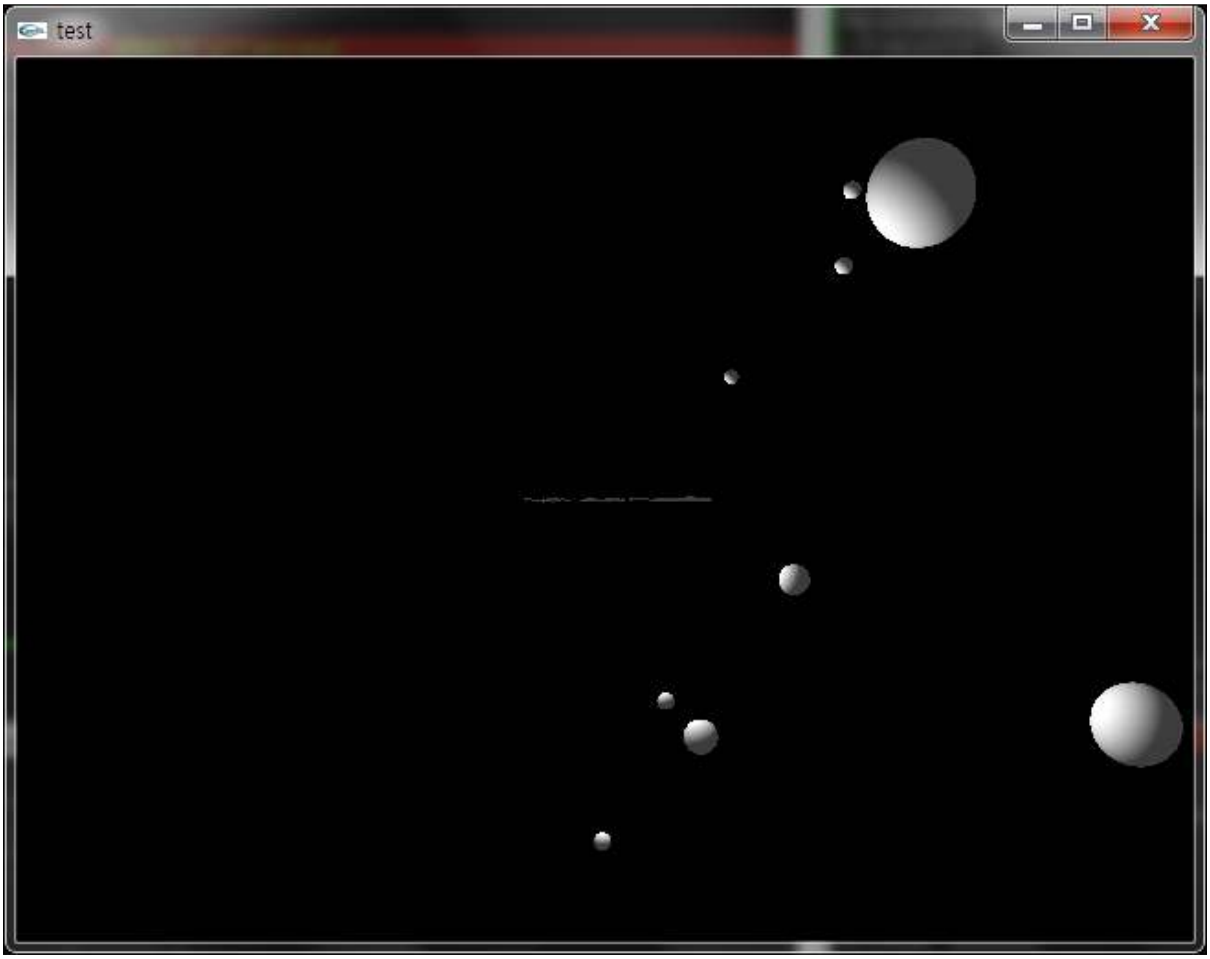


IAT 테이블을 살펴보면 gluLookAt, gluPerspective 함수가 win32.dll안에 있다고 하지만, 접두사가 gl인 함수들은 OpenGL 관련 함수들이다.
검색을 통해 win32.dll은 glu32.dll로, ws2_32.dll은 glut32.dll로 수정하였다.
DLL들을 수정 후 프로그램을 실행시키면 다음과 같이 OpenGL로 3D 태양계를 보여준다.
과거 Direct3D를 이용하여 비슷한 문제를 제작해본 경험이 있어서 태양과 행성의 위치들을 변경해 보기로 했다.

glTranslatef 함수는 축을 이동시키며 glutSolidSphere 함수는 구를 그린다.
따라서 태양의 위치를 이동시키기 위해 glTranslatef 함수의 인자를 조작하였다.
태양이 첫 번째로 그려지므로 처음 호출되는 glTranslatef 함수의 인자를 조작하였다.
FST를 FSTP로 수정하여 반대 값이 들어가도록 하였다.

```
00401483  .  83EC 0C       SUB ESP,0C
00401486  .  D95424 08     FST DWORD PTR SS:[ESP+8]
0040148A  .  D95424 04     FST DWORD PTR SS:[ESP+4]
0040148E  .  D91C24        FSTP DWORD PTR SS:[ESP]
00401491  .  FF15 A0E1410( CALL DWORD PTR DS:[<&OPENGL32,glTransla  OPENGL32,glTranslatef
```

그러면 다음과 같이 태양은 보이지 않는 곳으로 이동되고 태양이 있던 자리에 작게 무엇인가가 보이는 것을 확인할 수 있다.



뒤집혀져 있어서 보이지 않으므로 gluLookAt 함수의 인자를 조작하여 카메라의 위치를 이동시켰다.

gluLookAt의 eyez에 해당하는 세 번째 인자를 0x40140000000000000에서 0x4050000000000000으로 수정하였다.

그러자 다음과 같이 글자가 좀 더 명확히 보이는 것을 확인 할 수 있다.

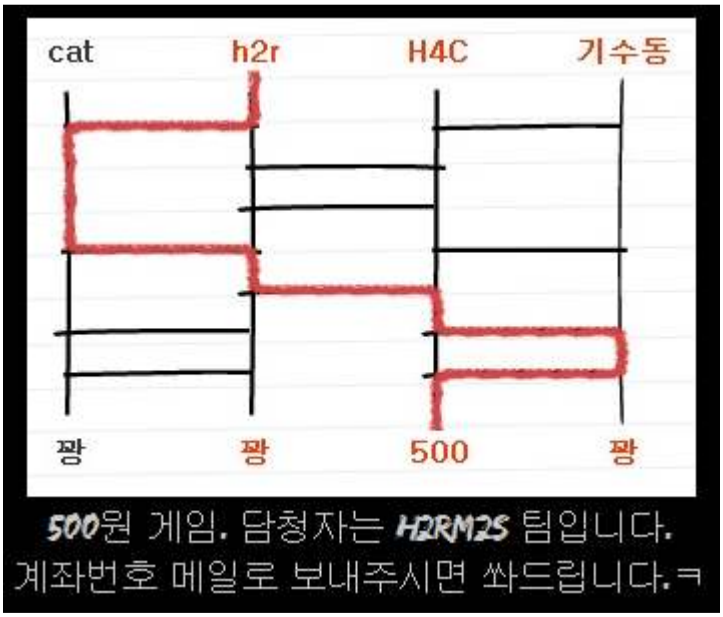
뒤집혀진 글자를 읽어보면 SATUrATION 이라는 것을 알 수 있다.

Key : SATUrATION

3. 후기

ARGOS 해킹 대회는 이번 참가가 처음이다.
작년도 문제를 보니 리버싱 문제들이 많아서 리버싱을 중점으로 공부하고 대비했으나 여러 분야들의 문제들이 나왔고 모두 흥미로웠다.
비록 고등학교 4명으로 참가했으나 최선을 다했고 꽤 높은 순위를 기록했다.
첫 날까지는 순위권에 들어서 좋았으나 둘째 날에 급격하게 순위가 하락하였다.
이번 대회를 통해 우리 팀의 취약한 분야를 알게 되었고 그 분야에 대해 좀 더 노력해야겠다고 생각했다.
학생들도 이루어진 팀이고 팀명도 없는 팀이지만 대회들에 참가하며 꾸준히 노력하고 있다.
내년 ARGOS 대회에서는 좀 더 좋은 성적을 거뒀으면 한다.
다음과 같이 에필로그에 팀 후기를 올렸었다.

알고 보니 후기를 올린 팀들 중 추첨을 통해 500원을 준다고 했었다고 한다.
다음과 같이 4:1의 경쟁률을 뚫고 추첨되어 500원을 받았다.



색다른 이벤트에 당첨되어 신기했었고, 내년 대회까지 더욱 열심히 공부해야겠다.