제1회 청소년 화이트해커 경진대회

3위 고기완(SelllSonic**)**

Level 1

1. 20A9(16진수)와 1100111111(2진수)의 합을 10진수로 나타내시오
2. 10011000과 00110101의 xor 연산을 하고 10진수로 나타내시오
3. N e w H e a r t

각각의 문자하나를 ascii 코드값의 10진수 합으로 나타내면?

1, 2, 3번 키를 붙여서 인증

1. 8361 + 831 = 9192
2. 152 xor 53 = 173
3. 78 + 101 + 119 + 72 + 101 + 97 + 114 + 116 = 798

Key) 9192173798

Level 2

주어진 웹페이지에 접속하면 '핸드폰으로 접속하세요' 라고 적혀있다.
모바일 브라우저로의 접속을 판단할 때 User-Agent를 체크하므로 User-Agent을 Mobile로 수정하여 접속하면 된다.

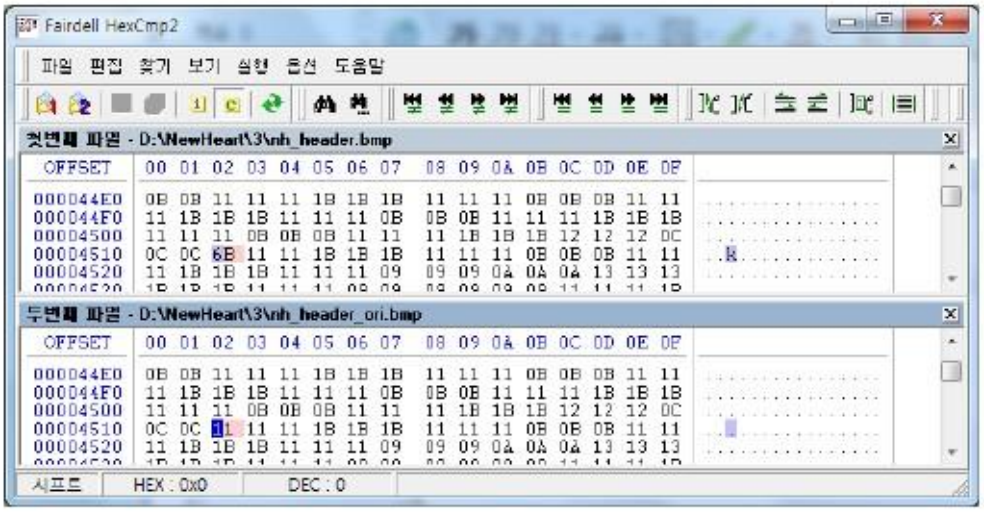
Key) Well begun is half done.

Level 3

누군가 뉴하트 홈페이지에 있는 로고에 비밀번호를 숨겨놓았다. 비밀번호를 알아내라.

주어진 비트맵 이미지를 확대하여 살펴보면 눈에 뜨이는 점이 몇 개가 있다.

이미지 파일의 일부가 수정되어 있으므로
뉴하트 홈페이지에 있는 원본 이미지와 바이너리 비교하여 차이나는 부분을 수집한다.



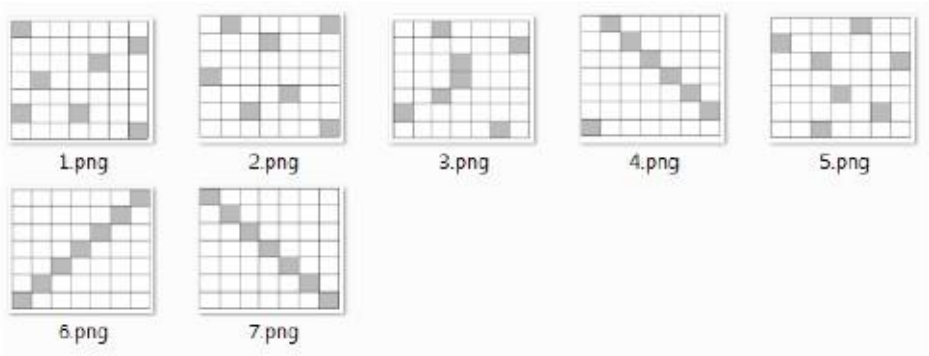
6B 65 79 69 73 6E 65 77 68 65 40 72 74 21 21

문자열로 변환 keyisnewhe@rt!!

Key) newhe@rt!!

Level 4

<decodebase85~aa$dEtfs+x&&hj+df#kgl*sfgmldvbzn@d/
~GiagsAQT#hhsso@tr;TwM$Y(le^ojs^4dawh?&ftzlwrgplY
)hOes!d*jgttmR^^zgngfdlxBdb~!!Op@cxuat}svf)vmst!z
1ropdtr&*Dity3c)3duaev*cvRtsb&4zt0dnads8hd@lk^jad
1879rwe#d$#ytr./dsudr>m^&ifg?bnBoahcv&(p4jxz#*lkt
d@#sda7zxcvb^^8o$aF1%7hg*23elj)hjklk,dfbfp>mnb,mo
@aytkotz;r((udx&I*dtkqae)0tl%^xr18;wvcvd~jbnzdfg>



주어진 문자열의 한 줄을 7x7 블록 이미지에 알맞게 7개로 분할한다.
<decode base85~ aa$dEtf s+x&&hj +df#kgl *sfgmld vbzn@d/

이미지에서 색칠된 부분에 해당하는 인덱스의 문자열만 추출하여 모은다.
<decode
base85~
aa$dEtf
s+x&&hj
+df#kgl
*sfgmld
vbzn@d/

<~E+*g/

위 과정을 반복해서 이어서 Base85 디코딩 하면 된다.

<~E+*g/GAhM4?YORgBOu!rDdR0d@r#drB4#7^F*),>@;I)1~>
password_is_hello_hacking_festival!!

Key) hello_hacking_festival!!

Level 5

h4ck.apk 파일의 압축을 풀어 classes.dex 파일을 dex2jar를 이용해 jar 파일로 변환하고, jd-gui 로 디컴파일한다.

```
public void df23089ikgdf()
{
  String str = decript("ygbahi?+hih5vrhhsb1r");
  Toast.makeText(this, str, 1).show();
}

private String decript(String paramString)
{
  String str1 = paramString;
  int[] arrayOfInt = new int[45];
  arrayOfInt[0] = 1;
  arrayOfInt[2] = 34;
  arrayOfInt[3] = 24;
  arrayOfInt[4] = 17;
  arrayOfInt[5] = 11;
  arrayOfInt[6] = 18;
  arrayOfInt[7] = 21;
        ...
  arrayOfInt[43] = 32;
  arrayOfInt[44] = 39;
  int i = 0;
  int j = str1.length();
  if (i >= j)
    return "";
  int k = 0;
  while (true)
  {
    if (k >= 45);
    label391:
    while (true)
    {
      i += 1;
      break;
      int m = str1.charAt(i);
      int n = "abcdefghijklmnopqrstuvwxyz/.1234567890-_:%+=?".charAt(k);
      if (m != n)
        break label393;
      int i1 = 0;
      while (true)
      {
        if (i1 >= 45)
          break label391;
        if (arrayOfInt[i1] == k)
        {
          String str2 = String.valueOf("");
          StringBuilder localStringBuilder = new StringBuilder(str2);
          char c = "abcdefghijklmnopqrstuvwxyz/.1234567890-_:%+=?".charAt(i1);
          String str3 = c;
          break;
        }
        i1 += 1;
      }
    }
    label393: k += 1;
  }
}
```

decript 함수에서 ygbahi?+hih5vrhhsb1r를 인자로 받아 어떤 연산을 하고 있다.

디컴파일이 올바르게 수행되지 않아 소스 분석이 힘들 수도 있겠지만 전체적인 흐름을 보면 인자로 전달 받은 문자가 “abcdefghijklmnopqrstuvwxyz/.1234567890~_:%+=?” 의 몇 번째 인덱스인가 찾고, arrayOfInt에서 그 인덱스에 해당하는 문자를 출력하면 된다.

그리고 디컴파일 결과에서 arrayOfInt[1]는 정의되어 있지 않은데 0으로 초기화 되어있어서 그런 것이다.

C언어로 비슷하게 구현해서 답을 얻을 수 있었다.

```
char input[] = "ygbahi?+hih5vrhhsb1r";
char c[] = "abcdefghijklmnopqrstuvwxyz/.1234567890~_:%+=?";
int arrayOfInt[45];
int i, j, n;

arrayOfInt[0] = 1;
arrayOfInt[1] = 0;
arrayOfInt[2] = 34;
arrayOfInt[3] = 24;
arrayOfInt[4] = 17;
arrayOfInt[5] = 11;
...
arrayOfInt[36] = 33;
arrayOfInt[37] = 4;
arrayOfInt[38] = 27;
arrayOfInt[39] = 19;
arrayOfInt[40] = 15;
arrayOfInt[41] = 12;
arrayOfInt[42] = 42;
arrayOfInt[43] = 32;
arrayOfInt[44] = 39;

for (i=0; i<(int)strlen(input); i++) {
        for (j=0; j<(int)strlen(c); j++) {
                if (input[i] == c[j]) {
                        for (n=0; n<45; n++) {
                                if (arrayOfInt[n] == j) printf("%c", c[n]);
                        }
                        break;
                }
        }
}
```

Key) diablo3+lol=hellgate

Level 6

nhf3.xap xap 파일은 윈도우폰 앱 형식이다. 압축을 풀어 nhf3.dll를 .Net Reflector 로 디컴파일한다.

button1 ~ button9 버튼 컨트롤은 arr 배열에서 자신의 번호에 해당하는 인덱스를 'index' 변수 값으로 채우고 'index' 변수를 1 증가시킨다.
다시 말해 버튼을 클릭한 순서대로 arr[x]에 해당하는 값은 1 ~ 9가 들어가게 된다.

```
private void button1_Click(object sender, RoutedEventArgs e)
{
    if (this.arr[1] == 0)
    {
        this.arr[1] = this.index;
        this.index = (byte) (this.index + 1);
        this.PageTitle.set_Text(this.PageTitle.get_Text() + this.button1.get_Content().ToString());
    }
}
```

그리고 Summit 버튼은 "a5b8f53248a781e32e4fac5190dbfabc"은 Password로,
arr을 Salt로 사용하여 "e069836af6c41b560477d80ce8b08a36"를 AES 암호화 하고 Base64 인코딩하여 화면에 출력한다.

```
private void Summit_Click(object sender, RoutedEventArgs e)
{
    Rfc2898DeriveBytes bytes = new Rfc2898DeriveBytes("a5b8f53248a781e32e4fac5190dbfabc", this.arr);
    Aes aes = new AesManaged();
    aes.Key = bytes.GetBytes(aes.KeySize / 8);
    aes.IV = bytes.GetBytes(aes.BlockSize / 8);
    using (MemoryStream stream = new MemoryStream())
    {
        ICryptoTransform transform = aes.CreateEncryptor();
        using (CryptoStream stream2 = new CryptoStream(stream, transform, CryptoStreamMode.Write))
        {
            byte[] buffer = Encoding.UTF8.GetBytes("e069836af6c41b560477d80ce8b08a36");
            stream2.Write(buffer, 0, buffer.Length);
            stream2.FlushFinalBlock();
        }
        this.PageTitle.set_Text(Convert.ToBase64String(stream.ToArray()));
    }
}
```

여기서 버튼의 클릭에 따라 달라지는 arr의 값을 알지 못하기 때문에 찾아야 한다.

압축 푼 파일 중에서 WMAppManifest.xml을 살펴보면 Description에 버튼의 순서가 보인다.

```
:.normal"  Author="newheart author" Description="134628957" Publis
```

버튼의 순서를 134628957로 하면 된다.

그런데 버튼의 순서가 버튼 Name이 아닌 buttonX_Click 이벤트의 디컴파일 결과에서 보이듯이 화면에 출력되는 Content를 기준으로 해야 한다.

nhf3.dll파일 내용을 살펴보면 아래와 같이 버튼 Name에 해당하는 Content가 존재한다.

```
<Button Content="3" Height="169" HorizontalAlignment="Left" Margin="0,6,0,0" Name="button1" Ver
<Button Content="5" Height="169" HorizontalAlignment="Left" Margin="154,6,0,0" Name="button2" V
<Button Content="6" Height="169" HorizontalAlignment="Left" Margin="304,6,0,0" Name="button3" V
<Button Content="4" Height="169" HorizontalAlignment="Left" Margin="0,181,0,0" Name="button4" V
<Button Content="8" Height="169" HorizontalAlignment="Left" Margin="154,181,0,0" Name="button5"
<Button Content="1" Height="169" HorizontalAlignment="Left" Margin="304,181,0,0" Name="button6"
<Button Content="2" Height="169" HorizontalAlignment="Left" Margin="0,356,0,0" Name="button7" V
<Button Content="7" Height="169" HorizontalAlignment="Left" Margin="154,356,0,0" Name="button8"
<Button Content="9" Height="169" HorizontalAlignment="Left" Margin="304,356,0,0" Name="button9"
```

Content가 1346289 57가 될 경우 button의 Name은 button6 button1 button4 button3 button7 button5 button9 button2 button8 이 된다.

즉 arr[6] = 1, arr[1] = 2, arr[4] = 3, arr[3] = 4, arr[7] = 5, arr[5] = 6, arr[9] = 7, arr[2] = 8, arr[8] = 9 가 된다.

해당 arr 배열에 대한 출력 값이 답이다.

```
arr[6] = 1;
arr[1] = 2;
arr[4] = 3;
arr[3] = 4;
arr[7] = 5;
arr[5] = 6;
arr[9] = 7;
arr[2] = 8;
arr[8] = 9;

Rfc2898DeriveBytes bytes = new Rfc2898DeriveBytes("a5b8f53248a781e32e4fac5190dbfabc", arr);
Aes aes = new AesManaged();
aes.Key = bytes.GetBytes(aes.KeySize / 8);
aes.IV = bytes.GetBytes(aes.BlockSize / 8);

using (MemoryStream stream = new MemoryStream())
{
    ICryptoTransform transform = aes.CreateEncryptor();
    using (CryptoStream stream2 = new CryptoStream(stream, transform, CryptoStreamMode.Write))
    {
        byte[] buffer = Encoding.UTF8.GetBytes("e069836af6c41b560477d80ce8b08a36");
        stream2.Write(buffer, 0, buffer.Length);
        stream2.FlushFinalBlock();
    }
    Console.WriteLine(Convert.ToBase64String(stream.ToArray()));
}
```

Key) u+Vscbgx4hX8Onbrk0dH8Rxcbdg1FnCOH8xn2Uy8aDkoUk4hcHvRK/LGpuMCqQ8N

Level 7

- 대회 서버가 닫혀있어 정확한 풀이를 쓸 수 없다.

주어진 페이지에 접속하면 룰렛 이벤트 페이지가 존재하며 한 아이피마다 30분에 한 번씩 룰렛을 돌릴 수 있고 0 ~ 3점을 랜덤으로 얻게 된다.
포인트를 얻으면 50점을 사용하여 힌트1을 열 수 있고, 100점을 사용하여 힌트2를 얻을 수 있다. 그리고 1000점을 모으면 답을 얻을 수 있다.

우선 웹페이지 소스를 분석해보면 dehydrate 자바스크립트 함수를 이용하여 스크립트 복호화를 진행한다.

복사해서 실행해 복호화를 하면 아래 소스가 나온다.
<script src=./roulette.js></script>
<script src=./aes.js></script>

roulette.js를 열어보면 XMLHttpRequest 함수를 이용하여
~~http://165.246.149.60/quiz/process.php 에 AES~~ 암호화하여 요청을 보내는 부분이 존재한다.
소스를 그대로 복사하여 timestamp에서 현재 시간을 조작하면 30분 제한을 우회하여 포인트를 계속 올릴 수 있다.

그런데 아이피당 100회 제한이 있으므로 최대 300포인트 까지만 올릴 수 있다.

300포인트로 힌트1을 보면 'timestamp' 라고 나오고,
힌트2를 보면 'UPDATE quiz set timestamp=(value), point=(value), counter=(value) WHERE ip_addr=(ip)' 라고 나온다.

힌트에서 알려준 쿼리를 바탕으로 SQL Injection을 하면 되겠다.

timestamp에 다음과 같이 인젝션 하여 포인트 조작이 가능하다.
timestamp=999999999999, point=1000, counter=7 WHERE ip_addr='x.x.x.x'#

Key) NewHeartBeat

Level 8

wavwav.wav 드라마 뉴하트 OST 음악 파일이 주어진다.

GoldWave로 열어서 Control창의 Spectrogram을 확인해보면 이상한 숫자들이 보인다.

333233423121341231 / 0001110011010000001000010101000110000001010 1

333233423121341231을 바탕으로 이진수들을 자리 수대로 나누면
000 111 001 10 100 000 0100 00 101 0 10 0 011 0000 0 01 010 1 이 되고,

0을 '.' 으로, 1을 '-' 으로 변경하여 모스 코드 디코딩을 하면 답이 된다.
... --- ..- -. -.. ... .-.. .. -.- . -. . .-- .... . .- .-. -

Key) soundslikenewheart

대회 기간 중에 가장 많은 시간을 소비한 문제인데 풀이는 여백이 많이 남네
여백여백여백 女白 여100

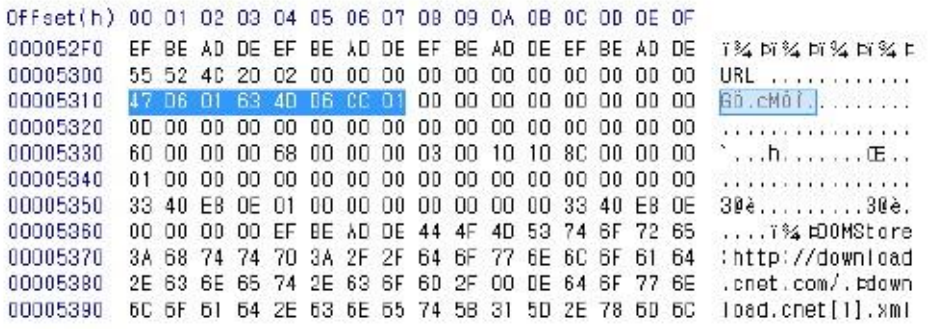
Level 9

Download Date?

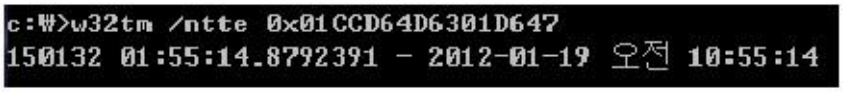
yyyyMMddHHmmss

파일을 열어보면 파일의 앞 부분이 아래와 같이 시작된다.
Client UrlCache MMF Ver 5.2

검색을 해보니 index.dat 파일이라는 것을 알 수 있었고,
URL Tag + 0x10 부분이 Last Accessed에 해당하는 부분이었다.



w32tm을 이용해 날짜 형식을 바꾸면 답을 구할 수 있다.



Key) 20120119105514

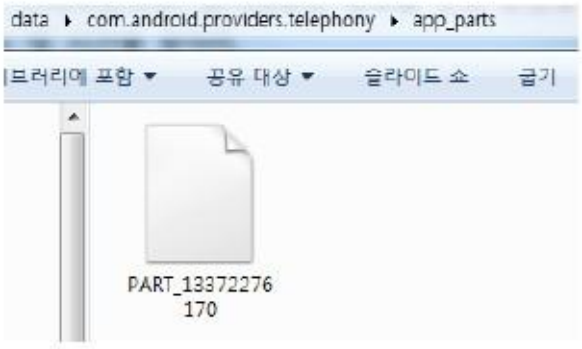
Level 10

NewHeart 수사대는 어떤 사건을 수사하던 중 마약 사건에 관련된 범인을 체포하였다. 범인은 마약을 밀거래하는 사람으로 특정일 특정장소에서 밀거래상과 접선할 예정이었다는 점을 자백하였으나 수사대는 더 이상의 자세한 내용은 밝혀내지 못했다. 유일한 단서는 범인이 가지고 있던 스마트폰으로, 암거래상과 정보를 주고 받았을 가능성이 높다. 암거래상과의 접선 장소 및 시간을 찾아라.

안드로이드 스마트폰 파일 전체를 덤프한 것처럼 보이는 압축 파일이 문제로 주어졌다.

문제 설명에서 스마트폰으로 암거래상과 정보를 주고받았다고 했을 때 문자 메시지를 이용했을 확률이 크므로 mms 이미지 첨부파일이 저장되는 경로에 가보면 이상한 파일이 있다.

data\data\com.android.providers.telephony\app_parts



PART_13372276170 파일의 시그니처를 보니 JPG 이미지 파일이므로 이미지뷰어로 확인하면 되겠다.

Key) IU_CONCERT_1800_PM_JUNE_02_2012

Level 11

newheart.sys 파일을 IDA로 열어서 분석한다.

0x11720 함수에서 ZwOpenFile 함수의 SSDT를 후킹 한다.
기존의 ZwOpenFile 함수 대신 새로 변경된 0x11540 함수에서는 파일 명이 C:\NewHeart\Memo.txt 일 때 파일에 내용을 쓰는 것으로 보인다.

```
int __stdcall sub_11540(int a1, int a2, int a3, int a4, int a5, int a6)
{
  ...
  if ( !stricmp(DestinationString.Buffer, "\\??\\C:\\NewHeart\\Memo.txt") )
  {
    RtlInitUnicodeString((PUNICODE_STRING)&v8, L"\\DosDevices\\C:\\NewHeart\\M
    ...
    ZwCreateFile(&Handle, 0xC0100000u, &ObjectAttributes, &IoStatusBlock, 0, 1
    sub_11180(&unk_13008);
    ZwWriteFile(Handle, 0, 0, 0, &IoStatusBlock, &byte_130E0, 0x18u, &ByteOff
    ZwClose(Handle);
```

sub_11180을 분석하면 되겠다.
인자로 전달되는 unk_13008은 0x13008에 해당하는 부분으로 0xD2, 0x04, 0x21, 0x4B, 0x38, 0xFD, 0x15 ... 알 수 없는 배열이다.

```
char *__stdcall sub_11180(const char *a1)
{
  unsigned int v1; // kr00_4@1
  char v3; // [sp+18h] [bp-30h]@1
  char v4; // [sp+19h] [bp-2Fh]@1
  ULONG_PTR v5; // [sp+40h] [bp-8h]@1
  int i; // [sp+44h] [bp-4h]@1
  int v7; // [sp+48h] [bp+0h]@1

  v5 = (unsigned int)&v7 ^ BugCheckParameter2;
  v1 = strlen(a1);
  v3 = 0;
  memset(&v4, 0, 0x27u);
  sub_11010(v1);
  for ( i = 0; i < (signed int)v1; ++i )
    *(&v3 + i) = byte_130A0[i] ^ a1[i];
  for ( i = 0; i < 40; ++i )
    byte_130E0[i] = *(&v3 + i);
  return &v3;
}
```

sub_11180 함수는 sub_11010을 호출하고, byte_130A0과 인자로 전달된 a1을 xor 하는 것이 전부이다.

sub_11010 함수는 복잡해 보이지만 디컴파일한 그대로 비슷하게 코딩하면 된다.
참고로 디컴파일 결과에서 보이는 dbl_13000는 0.200013 이다.

```
char            b130E0[40];

void sub_11010(int          a1)
{
        char
        double          v4[48] = {0};
        int             dd;
        bool                            i, v3;
                        v2;


            dd = 0.200013;

            for (i=0; i<10000; i++) {
                        dd = 4.0 * dd * (1.0 - dd);
            }


            v3 = -1;
            for (i=0; i<8*a1; i++) {
                        if (!(i%8)) v3++;
                        v4[v3] *= 2;
                        v2 = dd > 0.5;
                        v4[v3] = (char)v2 | v4[v3];
                        dd = 4.0 * dd * (1.0 - dd);
            }



            for (i=0; i<40; i++) {
                        b130E0[i] = v4[i];
            }

}

int main()
{
        unsigned charss[] = {0xD2, 0x04, 0x21, 0x4B, 0x38, 0xFD, 0x15, 0xAA, 0xBE, 0x3E,
0xD4, 0x6D, 0x93, 0xE7, 0xF3, 0x87, 0x03, 0x8E, 0xB4, 0x48, 0x29, 0x92, 0xDF, 0xAB, 0xCD, 0xFD,
0xFD, 0xFD, 0xFD, 0xDD, 0xDD, 0xDD, 0xF7, 0x37, 0x7F, 0x2F, 0x1C, 0xFF};
        charv3[43] = {0};
        inti, v1;


            v1 = (int)strlen((char*)ss);

            sub_11010(v1);

            for (i=0; i<v1; i++) {
                        v3[i] = b130E0[i] ^ ss[i];
            }


            printf("%s\n", v3);

            return 0;
}
```

실행하면 쓸모없는 부분까지 출력되므로 적절히 자르면 되겠다.

Key) Pocari_SWEAT

Level 12

prob.mov 400MB가 넘는 동영상 파일이 주어진다.

동영상을 감상 중 이상한 문자열이 휙휙 지나가 영상 편집 툴로 그 근처 프레임을 살펴봤다.

k(“TWVz”+ 라는 문자열이 20초(Frame 628)에 적혀있다.
30초(Frame 928)에는 "c2FnZ"+ 이 적혀있고, 50초 (Frame 1525)에는 "SA9IH"+이 있다.

00:20 k("TWVz"+
00:30 "c2FnZ"+
00:50 "SA9IH"+
01:20 "Ywbl9"+
02:10 "tc2cu"+
03:30 "cGhw")

k(TWVzc2FnZSA9IHYwbl9tc2cucGhw)
TWVzc2FnZSA9IHYwbl9tc2cucGhw
Base64 디코딩: Message = v0n_msg.php

v0n_msg.php 페이지에 접속해보면 아래와 같은 설명이 적혀있다.

> This is a hint message for this step. You will get the answer from the following formula. The 'f' is a function to help you find the key message. And the 'input' is an argument of the function. Finally output is a result of 'f(input)'. When 'f'. 'f' == 'v0n_f.html' and 'output' == 'v0n_output', FIND THE KEY FROM INPUT!

v0n_f.html의 obfuscation 함수에 어떠한 값을 넣어서 v0n_output 파일이 나온 것이므로 obfuscation 함수를 분석해야 한다.

그런데 자바스크립트가 난독화되어 있으므로 우선 TSGSUWUW 배열로 사용되는 함수 명을 치환해준다.

```
<html>
    <script>
        function obfuscation(XWRVFFRX) {
            OEXKQJIJ = "";
            NWNELNOE = "";
            AHVGFOAZ = "PK";
            VZVYGIGN = "MZ";
            SVRYPXOU = ["?," ? , "?," ? , "?," ? , "?," ? , "?," ? , "?," ? , "?," ? , "?," ? , "?," ? , "?," ? , "?," ? , "?," ? , "?," ?
, "?," ? , "?," ? , "?"];KDPFXBLF=[" ? , "?," ? , "?," ? , "?," ? , "?," ? , "?"];for(YVOAABVY=0;YVOAABVY<XWRVFFRX.length;YVOAABVY++)
{OOKYPEFF=TSGSUWUW[1](0xCC);if(XWRVFFRX.charCodeAt(YVOAABVY)>=2*TSGSUWUW[2](Math.PI)*TSGSUWUW[3](16)*3&&XWRVFFRX.charCodeAt(YVOAABVY)
<=TSGSUWUW[4](3*TSGSUWUW[5](Math.PI))*19)OEXKQJIJ=KDPFXBLF[TSGSUWUW[7](XWRVFFRX.charCodeAt(YVOAABVY))-48];else
OEXKQJIJ=XWRVFFRX[YVOAABVY];if(YVOAABVY%2)NWNELNOE=NWNELNOE+AHVGFOAZ+SVRYPXOU[TSGSUWUW[8](TSGSUWUW[6]()*31)]+OEXKQJIJ+TSGSUWUW[1]
(TSGSUWUW[8](TSGSUWUW[6]()*25)+65)+SVRYPXOU[TSGSUWUW[8](TSGSUWUW[6]()*31)]+OOKYPEFF+" - "+TSGSUWUW[1](TSGSUWUW[8](TSGSUWUW[6]
()*25)+65)+SVRYPXOU[TSGSUWUW[8](TSGSUWUW[6]()*31)]+TSGSUWUW[1](TSGSUWUW[8](TSGSUWUW[6]()*25)+65);else
NWNELNOE=NWNELNOE+VZVYGIGN+SVRYPXOU[TSGSUWUW[8](TSGSUWUW[6]()*31)]+OEXKQJIJ+TSGSUWUW[1](TSGSUWUW[8](TSGSUWUW[6]
()*25)+65)+SVRYPXOU[TSGSUWUW[8](TSGSUWUW[6]()*31)]+" - "+OOKYPEFF+TSGSUWUW[1](TSGSUWUW[8](TSGSUWUW[6]()*25)+65)+SVRYPXOU[TSGSUWUW[8]
(TSGSUWUW[6]()*31)]+TSGSUWUW[1](TSGSUWUW[8](TSGSUWUW[6]()*25)+65);}NWNELNOE = _$_$_$(NWNELNOE,90);return NWNELNOE;}TSGSUWUW=
[String.replace,String.fromCharCode,Math.sin,Math.sqrt,Math.abs,Math.cos,Math.random,Number,parseInt,String.Trim,String.concat];function
_$_$_$(_$, _$_){_$_$="
            ";for(_$_$_=0;_$_$_<_$.length;_$_$_++)_$_$=_$_$+TSGSUWUW[1](_$[_$_$_].charCodeAt()^_$_);return _$_$;}
    </script>

</html>
```

치환하고 나면 소스 분석이 훨씬 쉬워진다.

SVRYPXOU과 KDPFXBLF 배열 값이 ?로 나오는 것은 문자가 깨진 것이므로 헥스 에디터 등으로 확인하면 된다.

```
AHVGFOAZ = "PK";
VZVYGIGN = "MZ";
SVRYPXOU = ["?,"?,"?,"?,"?,"?,"?,"?,"?,"?,"?,"?,"?,"?,"?,"?,"?,"?,"?,"?,"?,"?,"?,"?,"?,"?,"?,"?,"?,"?,"?];
KDPFXBLF = ["?,"?,"?,"?,"?,"?,"?,"?,"?,"?"];
for (YVOAABVY = 0; YVOAABVY < XWRVFFRX.length; YVOAABVY++) {
    OOKYPEFF = String.fromCharCode(0xCC);
    if (XWRVFFRX.charCodeAt(YVOAABVY) >= 2 * Math.sin(Math.PI) * Math.sqrt(16) * 3 && XWRVFFRX.charCodeAt(
    Math.cos(Math.PI)) * 19) OEXKQJIJ = KDPFXBLF[Number(XWRVFFRX.charCodeAt(YVOAABVY)) - 48];
    else OEXKQJIJ = XWRVFFRX[YVOAABVY];
    if (YVOAABVY % 2) NWNELNOE = NWNELNOE + AHVGFOAZ + SVRYPXOU[parseInt(Math.random() * 31)] + OEXKQJIJ
    String.fromCharCode(parseInt(Math.random() * 25) + 65) + SVRYPXOU[parseInt(Math.random() * 31)] + OOK
    String.fromCharCode(parseInt(Math.random() * 25) + 65) + SVRYPXOU[parseInt(Math.random() * 31)] +
    String.fromCharCode(parseInt(Math.random() * 25) + 65);
    else NWNELNOE = NWNELNOE + VZVYGIGN + SVRYPXOU[parseInt(Math.random() * 31)] + OEXKQJIJ + String.from
    25) + 65) + SVRYPXOU[parseInt(Math.random() * 31)] + " - " + OOKYPEFF + String.fromCharCode(parseInt(Mat
    SVRYPXOU[parseInt(Math.random() * 31)] + String.fromCharCode(parseInt(Math.random() * 25) + 65);
}
```

위 소스에서 Math 함수가 사용된 부분은 고정 상수이다.

Math.abs(3 * Math.cos(Math.PI)) * 19는 0x39 이므로
인자로 전달된 문자가 숫자이면
OEXKQJIJ = KDPFXBLF[Number(XWRVFFRX.charCodeAt(YVOAABVY)) - 48]를 하고,
아니라면 OEXKQJIJ = XWRVFFRX[YVOAABVY] 하여 문자를 그대로 쓴다.

NWNELNOE에 내용을 추가하는 부분을 살펴보면 원본 문자열이 들어있는 OEXKQJIJ는 네 번째에 쓰여진다는 것을 알 수 있다.

NWNELNOE = NWNELNOE + VZVYGIGN(2byte) + SVRYPXOU[parseInt(Math.random() * 31)](1byte) + OEXKQJIJ

v0n_output에서 MZ 다음 다음 1Byte와 PK 다음 다음 1Byte를 모두 모은다.

74 68 F1 73 F1 73 74 68 65 63 68 F4 6C 6C F3 6E 67 F3 66 F0 72 79 F0 75 72 66 75 74 75 72 F3

F0 ~ F9는 위에서 봤듯이 입력받은 문자가 숫자인 경우 KDPFXBLF[X – 48] 한 것이다.

따라서 F0 ~ F9를 30 ~ 39로 바꿔서 문자열로 바꾸면 된다.

74 68 31 73 31 73 74 68 65 63 68 34 6C 6C 33 6E 67 33 66 30 72 79 30 75 72 66 75 74 75 72 33

Key) th1s1sthech4ll3ng3f0ry0urfutur3