# 사이버 가디언즈 챌린지

한세사이버보안고
1051p
cocoa1234
이진근

# Mic Check 1p

Mic Check

1 point

아~아~ 마이크 체크!

아래의 플래그를 입력하세요.

Flag = {CyberGuardiansChallenge}

# 난 해적왕이 될거야!! 100p

| NO | SAFES | OWNER |
|---|---|---|
| 3 | This is for my love Will | Elizabeth |
| 2 | Monkey Banana | undead mongkey |
| 1 | Don't tough my treasure! | Captain Jack |

13.125.21.90 | user

값

dW5kZWFkIG1vbmtleQ%3D%3D

undead monkey

Captain Jack

Q2FwdGFpbiBKYWNr

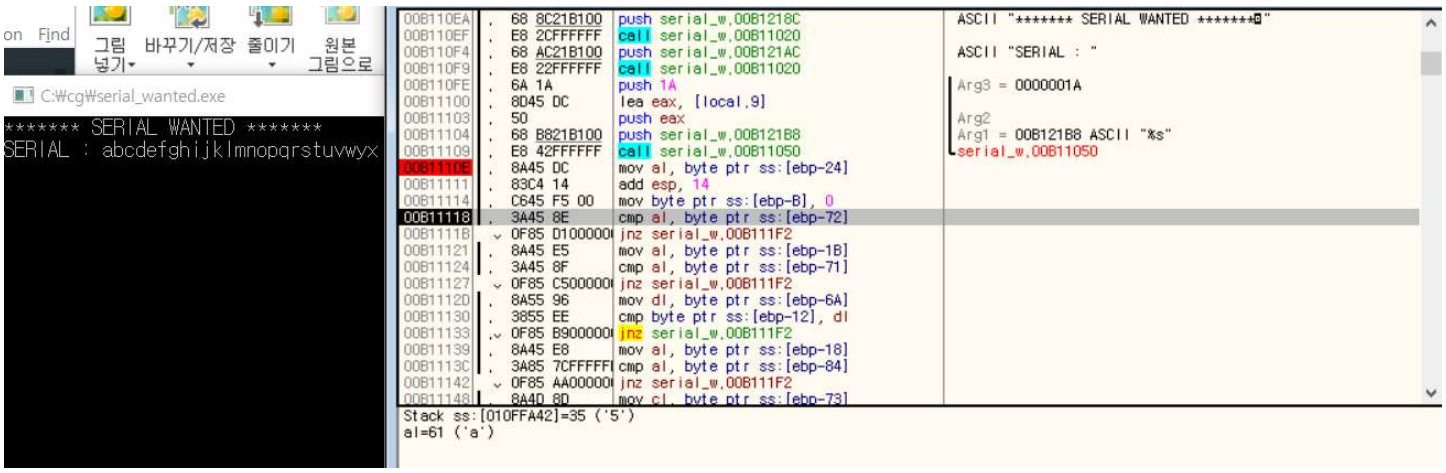
위처럼 Captain Jack이라는 문자를 base64로 encode하여 쿠키에 집어 넣은 후 글을 읽으면 jack.gold라는 파일을 다운받게 된다.

jack.gold라는 파일을 열면 플래그가 나온다

owner : Captain Jack Sparrow
Balance : 1,000,000,000
FLAG{Y0u_g0t_my_tr2asure_aga1n!}

# serial wanted 150p

```
00B110EA  .  68 8C21B100    push serial_w.00B1218C          ASCII "******* SERIAL WANTED *******"
00B110EF  .  E8 2CFFFFFF    call serial_w.00B11020
00B110F4  .  68 AC21B100    push serial_w.00B121AC          ASCII "SERIAL : "
00B110F9  .  E8 22FFFFFF    call serial_w.00B11020
00B110FE  .  6A 1A          push 1A                         Arg3 = 0000001A
00B11100  .  8D45 DC        lea eax, [local.9]
00B11103  .  50             push eax                        Arg2
00B11104  .  68 B821B100    push serial_w.00B121B8          Arg1 = 00B121B8 ASCII "%s"
00B11109  .  E8 42FFFFFF    call serial_w.00B11050          serial_w.00B11050
00B1110E  .  8A45 DC        mov al, byte ptr ss:[ebp-24]
00B11111  .  83C4 14        add esp, 14
00B11114  .  C645 F5 00     mov byte ptr ss:[ebp-B], 0
00B11118  .  3A45 8E        cmp al, byte ptr ss:[ebp-72]
00B1111B  .  0F85 D100000(  jnz serial_w.00B111F2
00B11121  .  8A45 E5        mov al, byte ptr ss:[ebp-1B]
00B11124  .  3A45 8F        cmp al, byte ptr ss:[ebp-71]
00B11127  .  0F85 C500000(  jnz serial_w.00B111F2
00B1112D  .  8A55 96        mov dl, byte ptr ss:[ebp-6A]
00B11130  .  3855 EE        cmp byte ptr ss:[ebp-12], dl
00B11133  .  0F85 B900000(  jnz serial_w.00B111F2
00B11139  .  8A45 E8        mov al, byte ptr ss:[ebp-18]
```




이렇게 입력하면 입력 값을 cmp로 비교를 한다. 처음에 길이 체크도 하는데 적당히 때려 맞춰주면 된다.

cmp로 비교를 하는데 문자들을 다 맞춰주면

5b'*4fDCB693#ad9a==49fFDb 라는 문자열이 나온다.

flag = 5b'*4fDCB693#ad9a==49fFDb

# hahaha 100p

```
  770 110.900182758 192.168.254.173   192.168.254.169   ICMP      81 Echo (ping) reply    id=0xeb05, seq=0/0, ttl=64
  772 110.903000944 192.168.254.169   192.168.254.173   ICMP     581 Echo (ping) reply    id=0xeb05, seq=2816/11, ttl=64
  773 110.903269630 192.168.254.169   192.168.254.173   ICMP     362 Echo (ping) reply    id=0xeb05, seq=3072/12, ttl=64
  776 110.907769356 192.168.254.169   192.168.254.173   ICMP     581 Echo (ping) reply    id=0xeb05, seq=1536/6, ttl=64
  777 110.907927369 192.168.254.169   192.168.254.173   ICMP     362 Echo (ping) reply    id=0xeb05, seq=1792/7, ttl=64
  806 139.960131861 192.168.254.173   192.168.254.169   ICMP      73 Echo (ping) reply    id=0xeb05, seq=0/0, ttl=64
  807 139.963259755 192.168.254.173   192.168.254.169   ICMP     109 Echo (ping) reply    id=0xeb05, seq=3328/13, ttl=64
  808 139.963465528 192.168.254.169   192.168.254.173   ICMP     109 Echo (ping) reply    id=0xeb05, seq=2048/8, ttl=64
  827 144.355590177 192.168.254.173   192.168.254.169   ICMP      74 Echo (ping) reply    id=0xeb05, seq=0/0, ttl=64
Internet Control Message Protocol
   Type: 0 (Echo (ping) reply)
   Code: 0
   Checksum: 0x2920 [correct]
   [Checksum Status: Good]
   Identifier (BE): 60165 (0xeb05)
   Identifier (LE): 1515 (0x05eb)
   Sequence number (BE): 3072 (0x0c00)
   Sequence number (LE): 12 (0x000c)
 Data (320 bytes)
      Data: 0000000000000000000000000000000000000000000000000000...
      [Length: 320]
```

```
0b0  20 69 73 20 75 73 65 64  20 69 6e 20 62 6f 74 68    is used  in both
0c0  20 77 69 72 65 64 20 6e  65 74 77 6f 72 6b 73 20    wired n etworks
0d0  61 6e 64 20 77 69 72 65  6c 65 73 73 20 63 6f 6d   and wire less com
0e0  6d 75 6e 69 63 61 74 69  6f 6e 73 5b 32 5d 2e 22   municati ons[2]."
0f0  0a 0a 73 6f 20 6d 79 20  6d 65 73 73 61 67 65 20   ..so my  message
100  69 73 20 22 4f 47 59 31  4e 54 5a 6c 59 54 4d 33   is "OGY1 NTZlYTM3
110  4d 57 51 79 59 54 4e 6c  4e 32 55 77 4e 6a 59 79   MWQyYTNl N2UwNjYy
120  5a 57 49 79 5a 54 42 68  5a 44 6c 68 4e 6d 55 33   ZWIyZTBh ZDlhNmU3
130  4e 44 6b 32 4d 57 49 32  4f 57 49 33 4d 54 4d 34   NDk2MWI2 OWI3MTM4
140  4d 47 49 33 4d 44 63 35  4d 7a 46 69 5a 6d 49 35   MGI3MDc5 MzFiZmI5
150  59 6d 4a 6d 4d 6a 63 7a  4e 67 3d 3d 22 0a 0a 2d   YmJmMjcz Ng=="..-
160  57 69 6b 69 70 65 64 69  61 0a                     Wikipedi a.
```

icmp 통신은 값이 다 보여서 잘 찾아보면 위와 같은 통신을 하는 것을 찾을 수 있다.

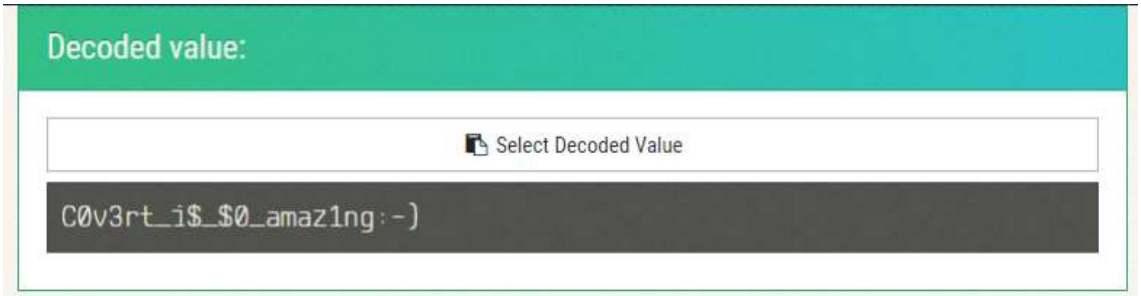
```
                        rt channel is mainly attributed to the computer networks, covert communication is a more general
word that is used in both wired networks and wireless communications[2]."

so my message is "OGY1NTZlYTM3MWQyYTNlN2UwNjYyZWIyZTBhZDlhNmU3NDk2MWI2OWI3MTM4MGI3MDc5MzFiZmI5YmJmMjczNg=="

-Wikipedia
```

=이 붙어 base64로 추측하고 decode하게 되면

8f556ea371d2a3e7e0662eb2e0ad9a6e74961b69b71380b707931bfb9bbf2736 라는 해쉬가 나오게 된다.

```
>>> len("8f556ea371d2a3e7e0662eb2e0ad9a6e74961b69b71380b707931bfb9bbf2736")
64
```

64바이트기 때문에 sha256을 추측할 수 있었고, 디코드 하면 아래와 같이 플래그가 나온다



# UUU 150p

```
AGGGUUUUUAGCAUGCCUCUUGAGGCCAGCGAAUUAAUAGUAGAGGAGGUGGAACGCUACGACGCUUAUGCA
UCUCACGCCCCCCCAUAUGAUGCUUACUAACGGGAACUUGCCACGGAG
```

그냥 이상한 평문인줄 알았더니 제목인 UUU로 한번 잘라보면

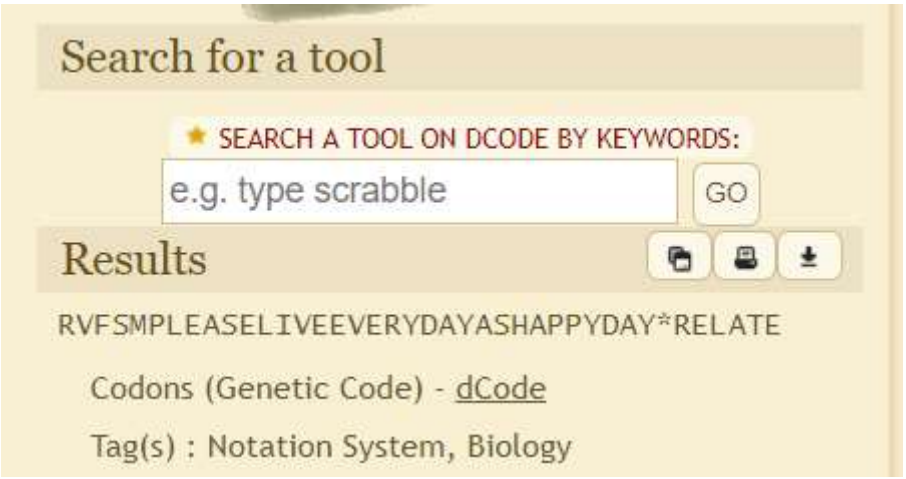
```
AGGG
UUU
UUAGCAUGCCUCUUGAGGCCAGCGAAUUAAUAGUAGAGGAGGUGGAACGCUACGACGCUUAUGCAUCUCACG
CCCCCCCAUAUGAUGCUUACUAACGGGAACUUGCCACGGAG
```

위가 4자리여서 3자리로 맞춰주자

```
AGG
GUU
UUU
AGCAUGCCUCUUGAGGCCAGCGAAUUAAUAGUAGAGGAGGUGGAACGCUACGACGCUUAUGCAUCUCACGCC
CCCCCAUAUGAUGCUUACUAACGGGAACUUGCCACGGAG
```

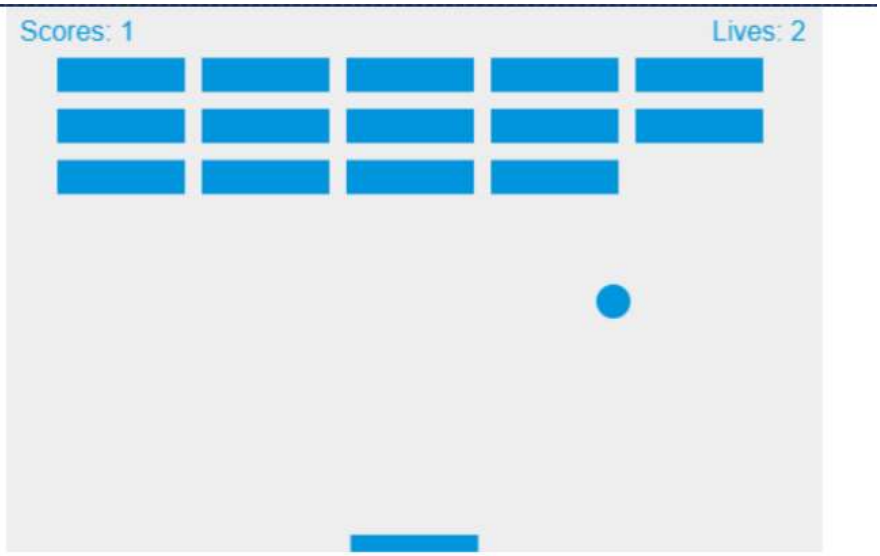
A U G C 4개의 알파벳만 있고 3자리씩 끊으니 dna rna 구조같았다.


검색하니 디코더가 나와서 복호화 해준다




RVFSMPLEASELIVEEVERYDAYASHAPPYDAY*RELATE 라는 문자열이 나오는데
인증하면서 때려 맞춰 풀었다.
flag = PLEASELIVEEVERYDAYASHAPPYDAY

# Why So Serious? 150p




```
<script>
    var e1="lo";var e2="rel";var e3="ion";var e4="ef";var e5="cat";var e6="oad";var e7="hr";var e8="hei";var e9="key";var e10="Code";var e11="Lis";var e12="th";var e13="Sty";var e14="Char";var e15="add";var e16="ght";var e17="le";var e18="fo";var e19="Eve";var e20="wid";var e21="from";var e22="nt";var e23="ner";var e24="nt";var e25="fill";var e26="te";var e27="ntB";var e28="tCo";var e29="ge";var e30="yId";var e31="nte";var e32="tEl";var e33="xt";var e34="eme";var e35="ntX";var e36="fse";var e37="ePa";var e38="tLe";var e39="tus";var e40="Text";var e41="arc";var e42="clie";var e43="rRe";var e44="rect";var e45="nPa";var e46="ft";var e47="sta";var e48="ct";var e49="begi";var e50="clos";var e51="of";var e52="clea";var e53="PI";var cn = document[e29+e32+e34+e27+e30]("myCanvas");var ctx=cn[e29+e28+e31+e33]("2d");var br=10;var x=cn[e20+e12]/2;var y=cn[e8+e16]-30;var dx=2;var dy=-2;var ph=10;var pw=75;var px=(cn[e20+e12]-pw)/2;var rp=false;var lp=false;var a1=5;var a2=3;var a3=75;var a4=20;var a5=10;var a6=30;var a7=30;var a8=0;var a9=3;var a10=[];for(c=0;c<a2;c++){a10[c]=[];for(r=0;r<a1;r++) {a10[c][r]={x:0,y:0,status:1};}}document[e15+e19+e24+e11+e26+e23](f2(82)+f2(92)+f2(64)+f2(93)+f2(86)+f2(78)+f2(87),f1,false);document[e15+e19+e24+e11+e26+e23](f2(82)+f2(92)+f2(64)+f2(76)+f2(73),f3, false);document[e15+e19+e24+e11+e26+e23](f2(84)+f2(86)+f2(76)+f2(74)+f2(92)+f2(84)+f2(86)+f2(79)+f2(92),f4,false);function f1(e){if(e[e9+e10]==39){rp=true;}else if(e[e9+e10]==37){lp=true;}}function f2(e){return String[e21+e14+e10](e-57);}function f3(e){if(e[e9+e10]==39){rp=false;}else if(e[e9+e10]==37){lp=false;}}function f4(e){var relativeX=e[e42+e35]-cn[e51+e36+e38+e46];if(relativeX>0&&relativeX <cn[e20+e12]){px=relativeX-pw/2;}}function f5(){for(c=0;c<a2;c++){for(r=0;r<a1;r++){var b=a10[c][r];if(b[e47+e39]==1){if(x>b.x&&x<b.x+a8&&y>b.y&&y<b.y+a4){dy=-dy;b[e47+e39]=0;a8++;if(a8>=a1*a2){var e54=((a8==a1*a2)?"f6()":"f7()");eval(e54);}}}}}}function f6(){alert(f2(96)+f2(118)+f2(108)+f2(25)+f2(110)+f2(112)+f2(119)+f2(21)+f2(25)+f2(122)+f2(118)+f2(119)+f2(126)+f2(107)+f2(120)+f2(109)+f2(106)+f2(24));document[e1+e5+e3][e2+e6]();}function f7(){document[e1+e5+e3][e7+e4]=f2(13)+f2(15)+f2(10)+f2(8)+f2(14)+f2(9)+f2(11)+f2(10)+f2(93)+f2(9)+f2(93)+f2(12)+f2(12)+f2(88)+f2(23)+f2(73)+f2(81)+f2(73);}function f8(){ctx[e49+e45+e12]();ctx[e41](x,y,br,0,Math[e53]*2);ctx[e25+e13+e17]="#0095DD";ctx[e25]();ctx[e50+e37+e12]();}function f9(){ctx[e49+e45+e12]();ctx[e44](px,cn[e8+e16]-ph,pw,ph);ctx[e25+e13+e17]="#0095DD";ctx[e25]();ctx[e50+e37+e12]();}function f10(){for(c=0;c<a2;c++){for(r=0;r<a1;r++){if(a10[c][r][e47+e39]==1){var brickX=(r*(a3+a5))+a7;var brickY=(c*(a4+a5))+a8;a10[c][r].x=brickX;a10[c][r].y=brickY;ctx[e49+e45+e12]();ctx[e44](brickX,brickY,a3,a4);ctx[e25+e13+e17]="#0095DD";ctx[e25]();ctx[e50+e37+e12]();}}}}function f11(){ctx[e18+e22]="16px Arial";ctx[e25+e13+e17]="#0095DD";ctx[e25+e40](f2(106)+f2(90)+f2(86)+f2(75)+f2(92)+f2(74)+f2(3)+f2(25)+a8,8,20);}function f12(){ctx[e18+e22]="16px Arial";ctx[e25+e13+e17]="#0095DD";ctx[e25+e40](f2(117)+f2(80)+f2(79)+f2(92)+f2(74)+f2(3)+f2(25)+a9,cn[e20+e12]-65,20);}function f13(){ctx[e52+e43+e48](0,0,cn[e20+e12],cn[e8+e16]);f10();f8();f9();f11();f12();f5();if(x+dx>cn[e20+e12]-br||x+dx<br){dx=-dx;}if(y+dy<br){dy=-dy;}else if(y+dy>cn[e8+e16]-br){if(x>px&&x<px+pw){dy=-dy;}else{a9--;if(!a9){alert(f2(126)+f2(120)+f2(116)+f2(124)+f2(25)+f2(118)+f2(111)+f2(124)+f2(107));document[e1+e5+e3][e2+e6]();}else{x=cn[e20+e12]/2;y=cn[e8+e16]-30;dx=3;dy=-3;px=(cn[e20+e12]-pw)/2;}}}if(rp&&px<cn[e20+e12]-pw){px+=7;}else if(lp&&px>0){px-=7;}x+=dx;y+=dy;requestAnimationFrame(f13);}f13();
</script>
```

canvas와 js를 이용한 게임이다.

보기 좋게 해주자

```
function f6() {
    alert(f2(96) + f2(118) + f2(108) + f2(25) + f2(110) + f2(112) + f2(119) + f2(21) + f2(25) + f2(122) + f2(118) + f2(119) + f2(126) + f2(107) + f2(120) + f2(109) + f2(106) + f2(24));
    document[e1 + e5 + e3][e2 + e6]();
}

function f7() {
    document[e1 + e5 + e3][e7 + e4] = f2(13) + f2(15) + f2(10) + f2(8) + f2(14) + f2(9) + f2(11) + f2(10) + f2(93) + f2(9) + f2(93) + f2(12) + f2(12) + f2(88) + f2(23) + f2(73) + f2(81) + f2(73);
}

function f8() {
    ctx[e49 + e45 + e12]();
    ctx[e41](x, y, br, 0, Math[e53] * 2);
    ctx[e25 + e13 + e17] = "#0095DD";
    ctx[e25]();
    ctx[e50 + e37 + e12]();
}

function f9() {
    ctx[e49 + e45 + e12]();
    ctx[e44](px, cn[e8 + e16] - ph, pw, ph);
    ctx[e25 + e13 + e17] = "#0095DD";
    ctx[e25]();
    ctx[e50 + e37 + e12]();
}
```

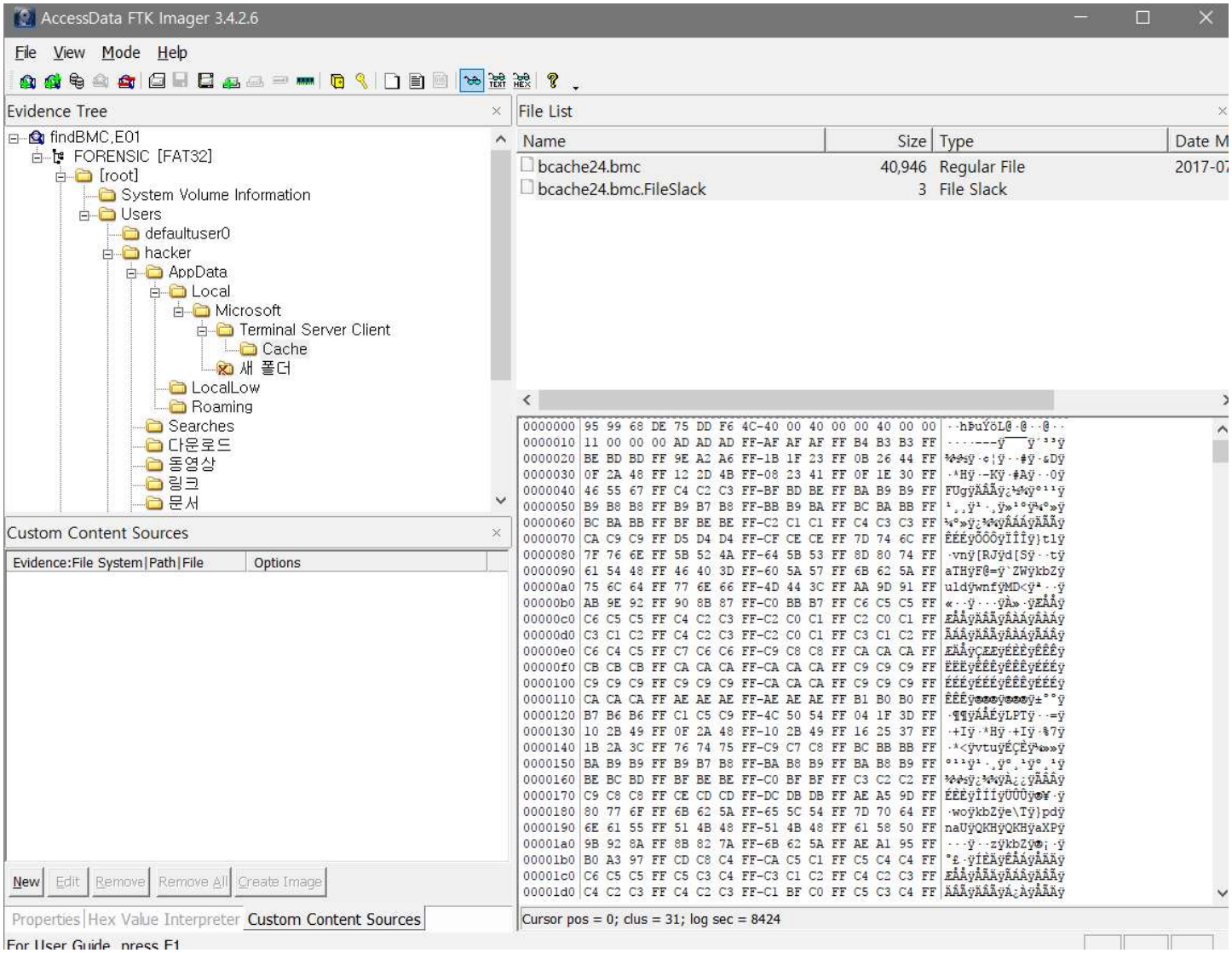
크롬 개발자도구 콘솔에서 함수들을 실행해 보았다

```
function f7() {
    document[e1 + e5 + e3][e7 + e4] = f2(13) + f2(15) + f2(10) + f2(8) + f2(14) + f2(9) + f2(11) + f2(10) + f2(93) + f2(9) + f2(93) + f2(12) + f2(12) + f2(88) + f2(23) + f2(73) + f2(81) + f2(73);
}
```

f6에서는 게임 클리어 함수, f7()함수에서 플래그가 실행되는 것을 알 수 있다.

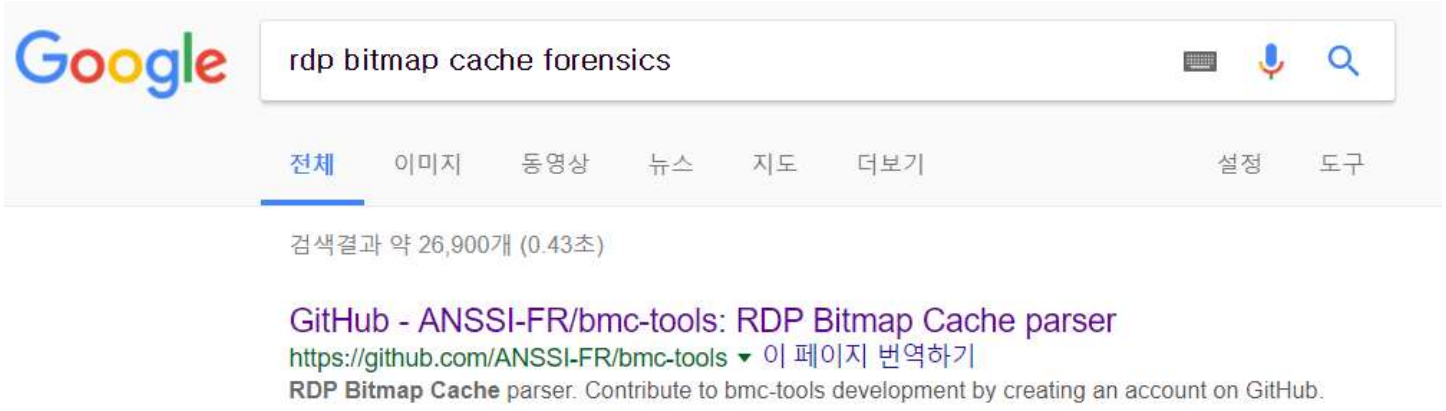
**Congrats! The flag is J4V4SCR1PT_1S_2ASY_T00_CR4CK**

# artifact 150p




bmc파일을 찾아보니 아래와 같다고 한다.

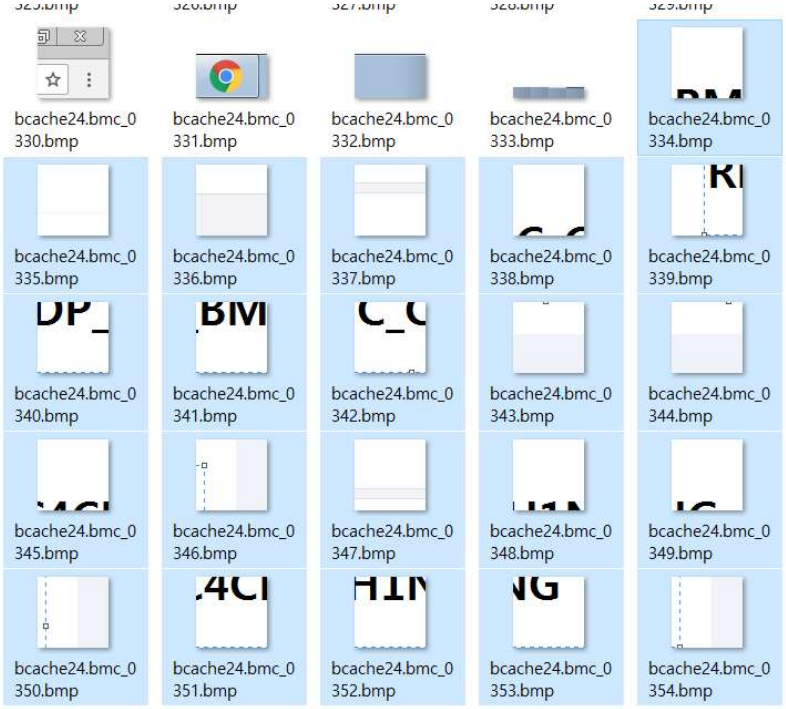
## .bmc 의 확장 파일 (Bitmap Cache File)




지문에서 주어지는 원격이라는 rdp와 위에서 얻은 bitmap cache를 조합해 검색하여 툴을 찾은 후

```
sori@tegra-ubuntu:~/bmc$ python bmc-tools.py -d /home/sori/bmc/resource/ -s bcache24.bmc
[+++] Processing a single file: 'bcache24.bmc'.
[===] Successfully exported 2556 files.
sori@tegra-ubuntu:~/bmc$ ls
bcache24.bmc  bmc-tools.py  resource
sori@tegra-ubuntu:~/bmc$ cd resource/
sori@tegra-ubuntu:~/bmc/resource$ ls
bcache24.bmc_0000.bmp  bcache24.bmc_0366.bmp  bcache24.bmc_0732.bmp  bcache24.bmc_1098.bmp  bcache24.bmc_1464.bmp  bcache24.bmc_1830.bmp  bcache24.bmc_2196.bmp
bcache24.bmc_0001.bmp  bcache24.bmc_0367.bmp  bcache24.bmc_0733.bmp  bcache24.bmc_1099.bmp  bcache24.bmc_1465.bmp  bcache24.bmc_1831.bmp  bcache24.bmc_2197.bmp
bcache24.bmc_0002.bmp  bcache24.bmc_0368.bmp  bcache24.bmc_0734.bmp  bcache24.bmc_1100.bmp  bcache24.bmc_1466.bmp  bcache24.bmc_1832.bmp  bcache24.bmc_2198.bmp
```

사용하면 이렇게 파일들이 나오게 된다




수상한 파일들만 따로 뽑아서 모양을 맞춰보면

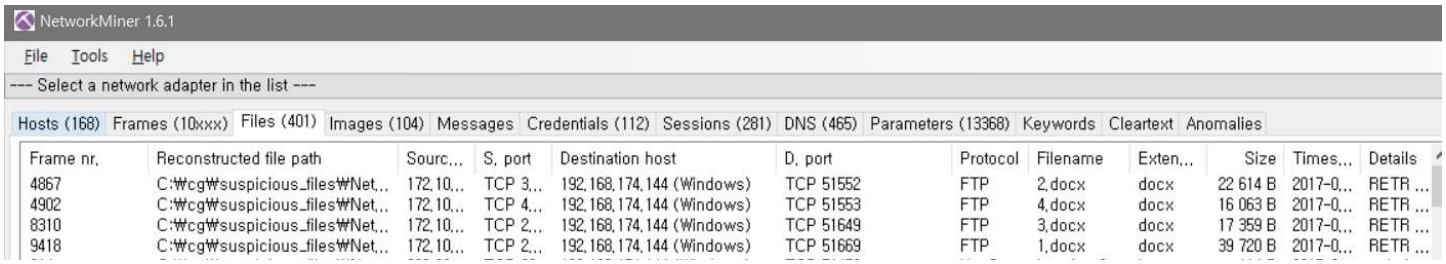
RDP_BMC_C4CH1NG 라는 문자가 나온다

# suspicious file 150p

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 4864 | 44.467149 | 192.168.174.144 | 172.104.106.149 | FTP | 67 | Request: RETR 2.docx |
| 4871 | 44.533764 | 172.104.106.149 | 192.168.174.144 | FTP | 121 | Response: 150 Opening BINARY mode data connection for 2.docx (22614 bytes). |
| 4892 | 44.599599 | 172.104.106.149 | 192.168.174.144 | FTP | 78 | Response: 226 Transfer complete. |
| 4893 | 44.618340 | 192.168.174.144 | 172.104.106.149 | FTP | 62 | Request: TYPE I |
| 4895 | 44.651662 | 172.104.106.149 | 192.168.174.144 | FTP | 85 | Response: 200 Switching to Binary mode. |
| 4896 | 44.652101 | 192.168.174.144 | 172.104.106.149 | FTP | 60 | Request: PASV |
| 4898 | 44.685990 | 172.104.106.149 | 192.168.174.144 | FTP | 108 | Response: 227 Entering Passive Mode (172,104,106,149,174,148). |
| 4899 | 44.686719 | 192.168.174.144 | 172.104.106.149 | FTP | 67 | Request: RETR 4.docx |
| 4912 | 44.753699 | 172.104.106.149 | 192.168.174.144 | FTP | 121 | Response: 150 Opening BINARY mode data connection for 4.docx (16063 bytes). |
| 4923 | 44.820607 | 172.104.106.149 | 192.168.174.144 | FTP | 78 | Response: 226 Transfer complete. |
| 8088 | 74.209725 | 192.168.174.144 | 172.104.106.149 | FTP | 62 | Request: TYPE A |
| 8109 | 74.242719 | 172.104.106.149 | 192.168.174.144 | FTP | 84 | Response: 200 Switching to ASCII mode. |
| 8301 | 79.676176 | 192.168.174.144 | 172.104.106.149 | FTP | 62 | Request: TYPE I |
| 8303 | 79.709139 | 172.104.106.149 | 192.168.174.144 | FTP | 85 | Response: 200 Switching to Binary mode. |
| 8304 | 79.709500 | 192.168.174.144 | 172.104.106.149 | FTP | 60 | Request: PASV |
| 8306 | 79.743424 | 172.104.106.149 | 192.168.174.144 | FTP | 108 | Response: 227 Entering Passive Mode (172,104,106,149,107,109). |
| 8307 | 79.744129 | 192.168.174.144 | 172.104.106.149 | FTP | 67 | Request: RETR 3.docx |
| 8323 | 79.810895 | 172.104.106.149 | 192.168.174.144 | FTP | 121 | Response: 150 Opening BINARY mode data connection for 3.docx (17359 bytes). |
| 8331 | 79.877252 | 172.104.106.149 | 192.168.174.144 | FTP | 78 | Response: 226 Transfer complete. |
| 9408 | 99.019488 | 192.168.174.144 | 172.104.106.149 | FTP | 62 | Request: TYPE I |
| 9410 | 99.053288 | 172.104.106.149 | 192.168.174.144 | FTP | 85 | Response: 200 Switching to Binary mode. |
| 9411 | 99.053605 | 192.168.174.144 | 172.104.106.149 | FTP | 60 | Request: PASV |
| 9413 | 99.086569 | 172.104.106.149 | 192.168.174.144 | FTP | 107 | Response: 227 Entering Passive Mode (172,104,106,149,96,186). |
| 9414 | 99.087352 | 192.168.174.144 | 172.104.106.149 | FTP | 67 | Request: RETR 1.docx |
| 9420 | 99.154679 | 172.104.106.149 | 192.168.174.144 | FTP | 121 | Response: 150 Opening BINARY mode data connection for 1.docx (39720 bytes). |

와이어샤크로 열어보면 ftp로 파일을 다운로드 한 목록을 볼 수 있다

네트워크 포렌식 툴인 Network-Miner로 파일을 내려 받을 수 있다.




저 파일들을 모두 열어보면

1.docx : VGtScVEzTkVWWGhL

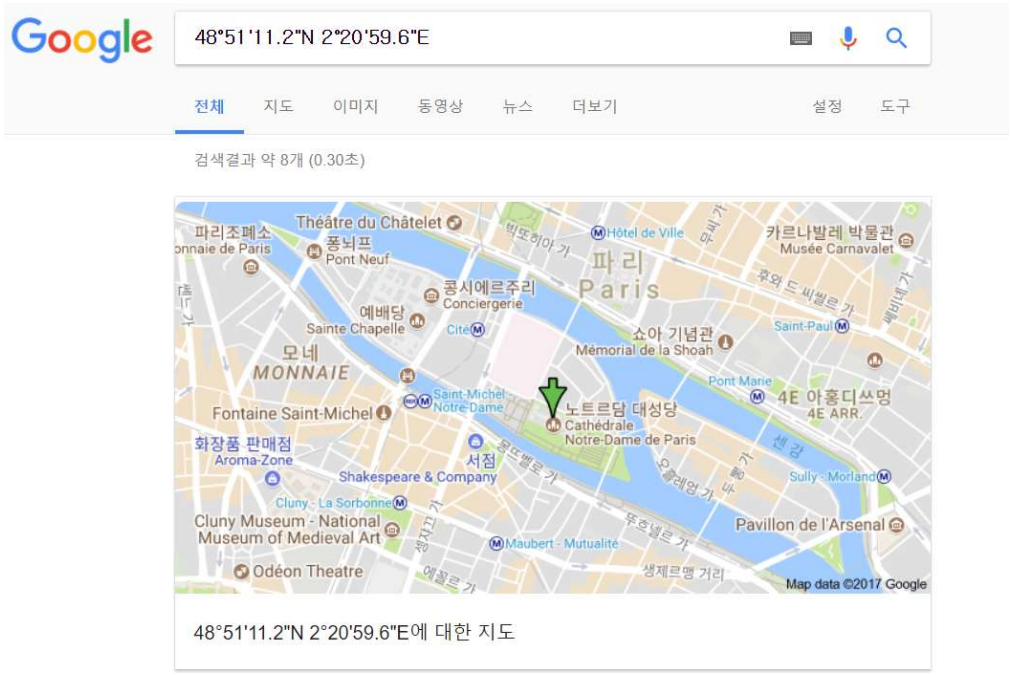
2.docx : ZWtWNFRHcEphVlJw

3.docx : UVhsM2NrRjVUVU5q

4.docx : TVU5VE5ESkphMVU5

네 개의 문자열을 합쳐서 base64로 계속 돌리면

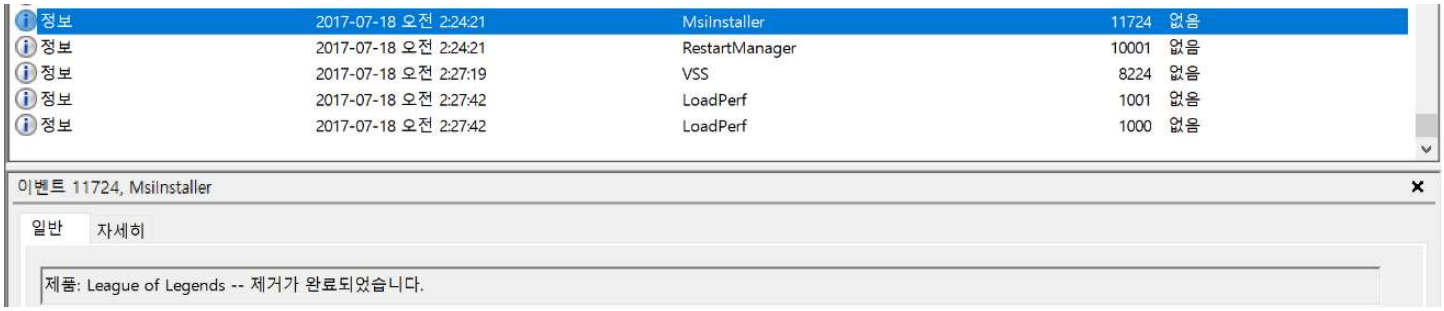
48°51'11.2"N 2°20'59.6"E 라는 좌표가 나오게 된다.




flag = cathedralenotredamedeparis

# mommy is detective! 100p

Application.evtx 파일을 열어 로그를 분석해보면 아래와 같이 나온다.





flag = 2017-07-14-18-15-36_2017-07-18-02-24-21