

순천향대학교 2011 청소년 정보보호 페스티벌 Write-Up

Xero

박준혁 (한국디지털미디어고등학교 1학년)

2011-10-16

wmsgurzxc@nate.com

[C] Trivial 100P

Subject

secu2011

Comment

0xC73B9452

0xC73B95C6

0xC73B95E6

By. MANO

C번은 간단한 문제였다.

0xC73B9452 값을 보고 10진수로 변환하니 3342570578 값이 나왔다.

최근에 ip를 10진수, 16진수로도 표현할 수 있다는 것을 알게 되어서 myip.kr에서 ip주소로 변환해 199.59.148.82 라는 주소를 얻었다.

들어가보니 트위터 사이트가 나타났고, 다른 주소들도 마찬가지로였다.

계속 고민하다가 subject가 secu2011 인것을 보고 <http://twitter.com/secu2011> 으
로 들어가보았다.

그러자 다음과 같이 키 값을 발견했고 인증에 성공했다.



Key : C0NV327_IP_4DD2355

[D] Trivial 100P

Subject

This is crypto

Comment

ThisIsCrypto

By. Rust


File : ThisIsCrypto¿.zip


Hint

1. 힌트파일 추가

다음과 같이 txt파일과 pdf파일이 주어진다.

pdf파일은 추가된 힌트파일이다.

 ThisIsCrypto.txt

 ThisIsKey¿.pdf

추가된 pdf파일의 내용은 다음과 같다.

To the Members of the California State Assembly:

I am returning Assembly Bill 1176 without my signature.

For some time now I have lamented the fact that major issues are overlooked while many unnecessary bills come to me for consideration. Water reform, prison reform, and health care are major issues my Administration has brought to the table, but the Legislature just kicks the can down the alley.

Yet another legislative year has come and gone without the major reforms Californians overwhelmingly deserve. In light of this, and after careful consideration, I believe it is unnecessary to sign this measure at this time.

Sincerely,

Arnold Schwarzenegger

위의 내용을 검색해 다음과 같은 사실을 알게 되었다.

캘리포니아 주의회 의원인 Tom Ammiano가 슈왈제네거에게 "Kiss my gay ass"라고 욕을 하곤 주지사 연설 도중 나간 일이 있다고 한다. 슈왈제네거는 그에 대한 대답으로 해당 의원이 낸 법안에 거부권을 행사하면서 다음과 같은 글을 썼다고 한다.

물론 내용은 문제될 게 없지만 일부러 세로로 욕을 쓴 것이다. 세로로 읽어보면 Fuck You 라는 글을 볼 수 있다.

txt 파일의 내용은 다음과 같다.

Dear My Friend,

The Keyword is close of we. Don't take it seriously.
(An end of This sentences, probably you can find a password.)
Are you a hacker with good skills?
Then do you have a good sense for guessing?
Or like spend time to try to find something?
It is a good chance for test your skill and sense!

Are You Ready? Ok. Let's go.
dEraoAsdfFvvZoeutQplmieVxsnisdEt f
lioOusNioewaTconOdeEssotNlfigssih
cHedAoxRcsmereWILPieTyheTeFioSmii
genlclyBsescHwteRustCnuPwithGredn
idiatidelerfosinsTlsooemlwmrRpsot
NdlqiShfkoYansAudWTfrExvUzpeestEi
hoAsvCoNiFetzOsesdnArwMsmhuenLths
rcInaESeuigrgronalPlPeaansurdylaS
FSLmtclDntGAlmiaFfwlAsotntFsotfge
sLoneFelnssehtoHMeuingndgHgexeotr
PasSThisistjustAWrongkeylmaliarlao
LuisrcMlsNhhlguecbgSfstraelstaid
tpmealnlnbifMeskwippqlzXTcraAhdrr
fcmpipiOntliHomunddgrShangasmEnti
agonCsostRvLdleptymbunnwoezxfgrip
mdGaaensiWlnpilfwibwdAwweadgvidul
srOwrdirhcmhguryPnoaeesMtolsswto
heUndlCenHhaasOsEaRpooPaipfrelnhl
Whoo! Are you know that means of that quotes?
This crypto algorithm is very very Strong! Good Luck For you!

by Rust :D

pdf 에서 세로로 비밀문장을 썼던 것을 생각하고 세로로 읽어보면 다음의 문장을 발견할 수 있고 인증에 성공하였다.

Dear My Friend,

The Keyword is close of we. Don't take it seriously.
(An end of This sentences, probably you can find a password.)
Are you a hacker with good skills?
Then do you have a good sense for guessing?
Or will you spend time to try to find something?
It is a good chance for test your skill and sense!

Are You Ready? Ok. Let's go.
dErsozsdFvVzoeutQplmieVxsnisdEt f
lioUuNioewaTconOdeEssotNlfigssih
cHeAxxRcsmereWlLPieTyheTeFioSmii
genccyBsescHwteRustCnuPwithGredn
idistdelerfosinsTlsooemlwmrRpsot
NdleiiShfkoVansAudWTfrExvUzpeestEi
hoAsvCoNiFetzOsesdnArwMsmhuenLths
rclicaeSeuigrgronalplPeaansurdylaS
FSLtclldntGAlmiaFfWlAsotntFsotfge
sLoeFeInssehtoHMeuingndgHgexeotr
PastThisisjustAWrongkeylmaliarlao
LuisrcMlsNhhlguecbgSfstraoelsTaid
tpmeaenlNbifMeskwippqlzXTcrcAhdrr
fcmppOnltliHomunddgrShangasmEnti
agorCsostRvLdleptymbunnwoezxfgrip
mdGaenslWlnpilfwibwdAwweadgvidul
srOvrlrhcmhguryPnoaeesMtolsswto
heUddcenHhaasOsEaRpooPaipfrelnhl
Whoa! Are you know that means of that quotes?
This crypto algorithm is very very Strong! Good Luck For you!

by Rust :D

Key : YouActivateTrapCard!

[E] Analysis 100P

Subject

♥0♥

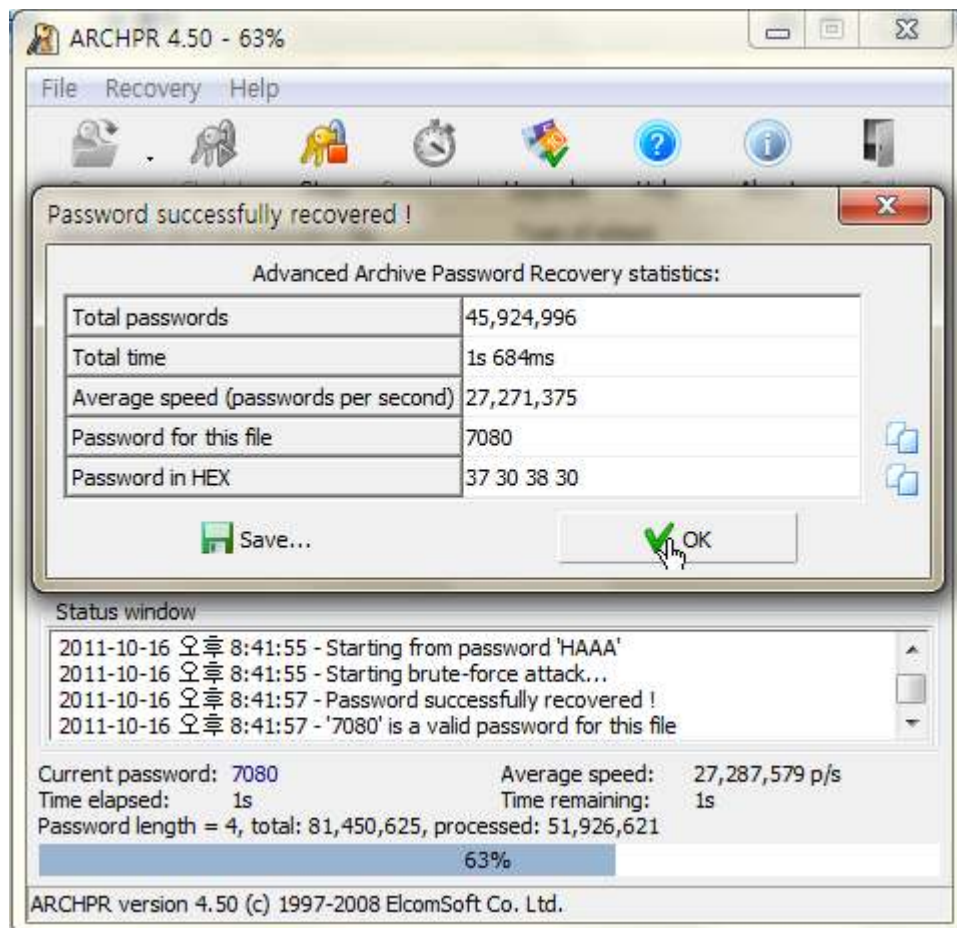
Comment

소느님 사랑해여♥0♥

By. Hak6a6y

File : wOw.zip

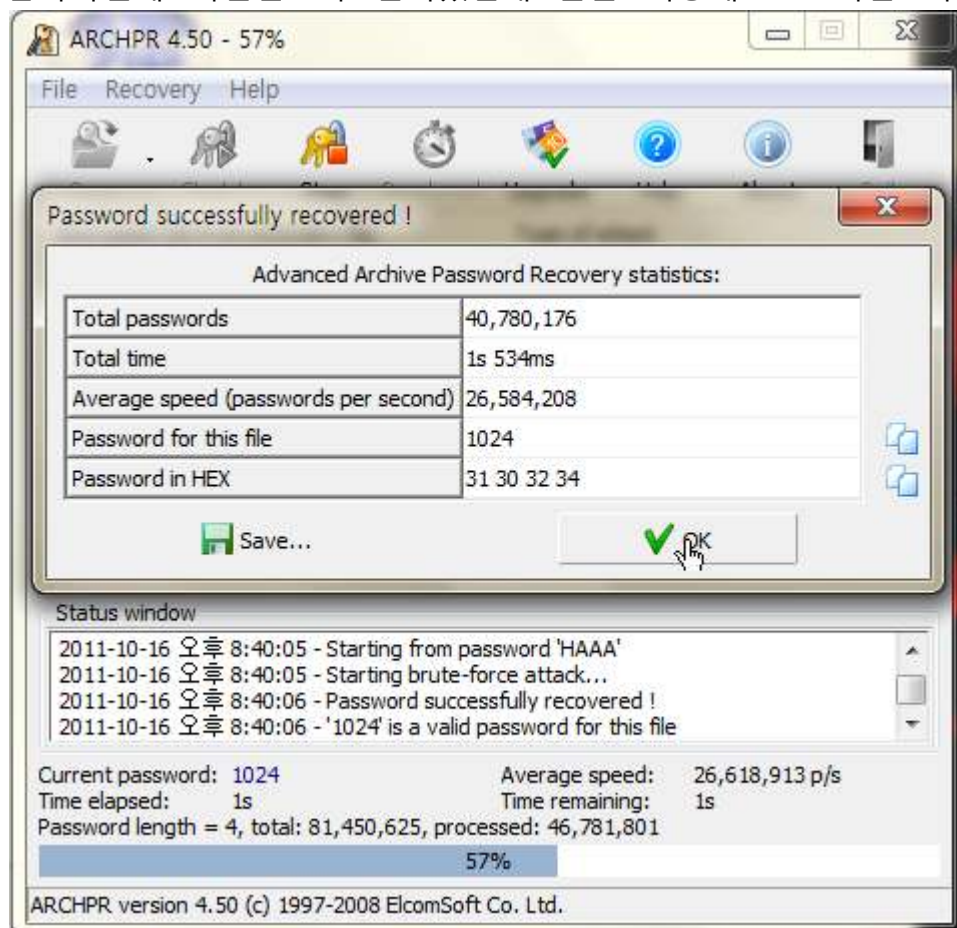
wOw.zip 파일에 압축 암호가 걸려있길래 ARCHPR 툴을 이용하여 7080이라는 비밀번호를 알아내었다.



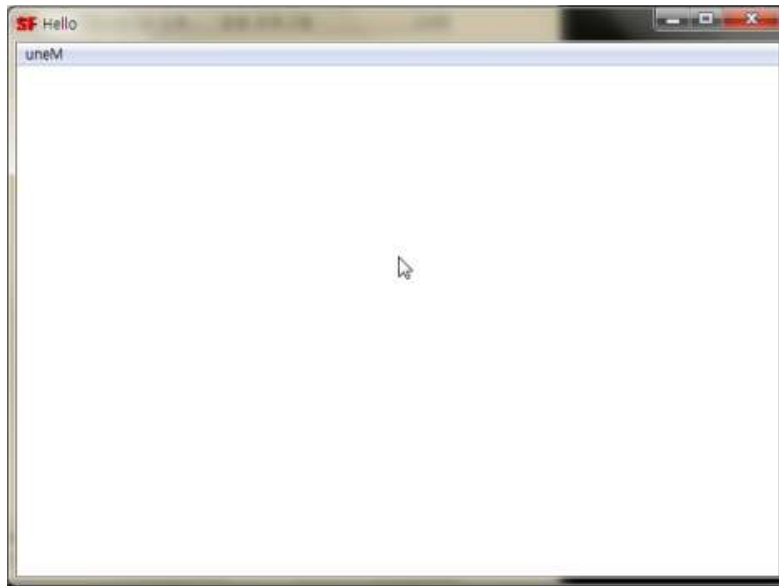
압축을 해제하니 다음과 같이 두 개의 이미지와 또 다른 압축 파일 하나를 주었다.



압축파일에 비밀번호가 걸려있길래 툴을 이용해 1024라는 비밀번호를 구했다.



압축을 해제하면 exe파일 하나가 있는데 실행하면 다음과 같이 창이 뜬다.

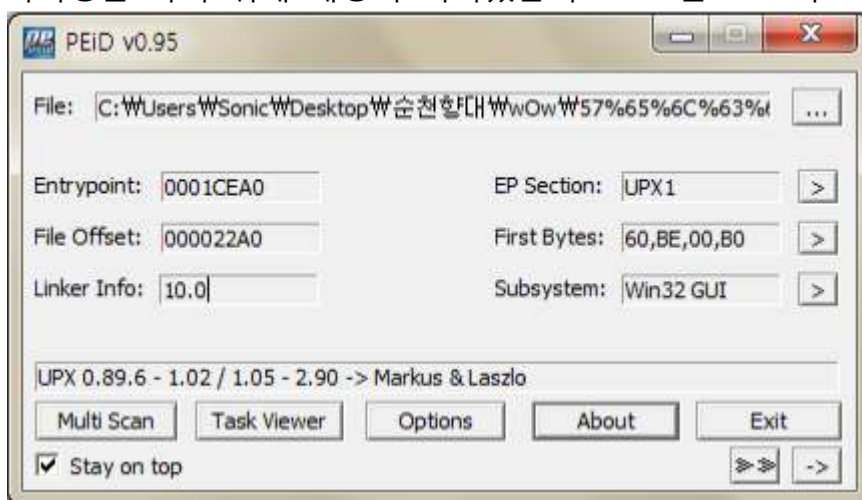


메뉴의 copyright를 누르면 다음과 같은 메시지창이 뜬다.



위의 값을 인증해보았으나 답이 아니었다.

리버싱을 하기 위해 패킹이 되어있는지 PEiD 툴로 보니 UPX 패킹이 되어있었다.



툴로 언패킹 해도 되지만 수동으로 언패킹 하기로 했다.

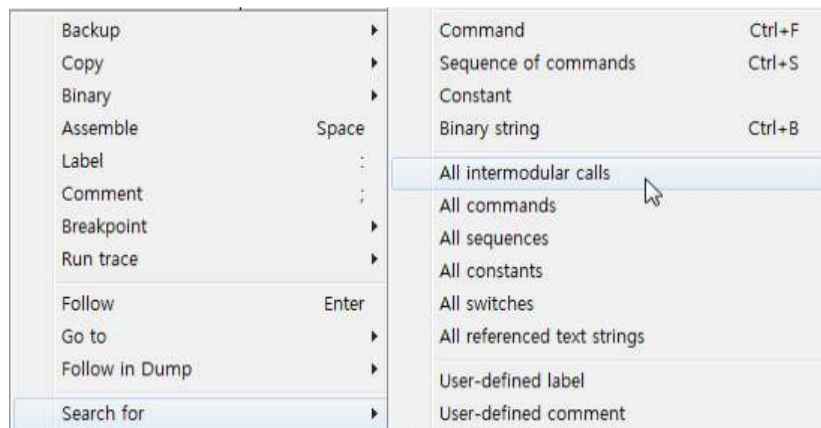
UPX 언패킹 방법은 어셈 코드의 맨뒤 DB 00 코드 세 줄 위의 JMP 구문에 F2를 눌러 브레이크를 걸고 F9로 실행하여 그 부분에서 멈추면 F8을 한번 누르면 패킹이 풀린 실제 코드부분으로 들어가게 된다.

0041CFE3	54	PUSH ESP
0041CFE4	50	PUSH EAX
0041CFE5	53	PUSH EBX
0041CFE6	57	PUSH EDI
0041CFE7	FFD5	CALL EBP
0041CFE9	58	POP EAX
0041CFEA	61	POPAD
0041CFEB	8D4424 80	LEA EAX,DWORD PTR SS:[ESP-80]
0041CFEF	> 6A 00	PUSH 0
0041CFF1	39C4	CMP ESP,EAX
0041CFF3	75 FA	JNZ SHORT Qnkgd2Fp,0041CFEF
0041CFF5	83EC 80	SUB ESP,-80
0041CFF8	- E9 8941FFFF	JMP Qnkgd2Fp,00411186
0041CFFD	00	DB 00
0041CFFE	00	DB 00
0041CFFF	00	DB 00

다음은 언패킹 후의 실제 코드부분이다.

00411186	↙ E9 450C0000	JMP	Qnkgd2Fp,004111D00	
0041118B	↙ E9 22290000	JMP	Qnkgd2Fp,00413AB2	JMP to MSVCR100,_controlfp_s
00411190	↙ E9 C5290000	JMP	Qnkgd2Fp,00413B5A	JMP to kernel32,GetSystemTimeAsFileTime
00411195	↙ E9 A2290000	JMP	Qnkgd2Fp,00413B3C	JMP to ntdll,RtlDecodePointer
0041119A	↙ E9 19290000	JMP	Qnkgd2Fp,00413AB8	JMP to MSVCR100,_invoke_watson
0041119F	↙ E9 2C230000	JMP	Qnkgd2Fp,004134D0	
004111A4	↙ E9 C7020000	JMP	Qnkgd2Fp,00411470	
004111A9	↙ E9 881A0000	JMP	Qnkgd2Fp,00412C36	JMP to MSVCR100,_CRT_RTC_INITW
004111AE	↙ E9 95290000	JMP	Qnkgd2Fp,00413B48	JMP to kernel32,GetTickCount
004111B3	↙ E9 38080000	JMP	Qnkgd2Fp,004119F0	JMP to USER32,KillTimer
004111B8	↙ E9 63210000	JMP	Qnkgd2Fp,00413320	
004111BD	↙ E9 A4290000	JMP	Qnkgd2Fp,00413B66	JMP to ntdll,RtlAllocateHeap
004111C2	↙ E9 E31E0000	JMP	Qnkgd2Fp,004130AA	JMP to MSVCR100,_amsg_exit
004111C7	↙ E9 12200000	JMP	Qnkgd2Fp,004131DE	JMP to MSVCR100,_XcptFilter
004111CC	↙ E9 43080000	JMP	Qnkgd2Fp,00411A14	JMP to USER32,SendMessageW
004111D1	↙ E9 20200000	JMP	Qnkgd2Fp,004131F6	JMP to MSVCR100,_CrtSetCheckCount
004111D6	↙ E9 01290000	JMP	Qnkgd2Fp,00413ADC	JMP to kernel32,InterlockedExchange
004111DB	↙ E9 50290000	JMP	Qnkgd2Fp,00413B30	JMP to kernel32,UnhandledExceptionFilter

Search for의 All intermodular calls 기능으로 호출되는 함수목록을 보았다.



다음과 같이 호출되는 함수들이 보인다.

Found intermodular calls		
Address	Disassembly	Destination
00411186	JMP Qnkgd2Fp,00411000	(Initial) CPU selection
004114A8	CALL DWORD PTR DS:[418284]	GD132.GetStockObject
004114C1	CALL DWORD PTR DS:[41844C]	USER32.LoadCursorW
004114D9	CALL DWORD PTR DS:[418450]	USER32.LoadIconW
00411512	CALL DWORD PTR DS:[418454]	USER32.RegisterClassW
00411554	CALL DWORD PTR DS:[418458]	USER32.CreateWindowExW
0041156E	CALL DWORD PTR DS:[41845C]	USER32.ShowWindow
00411587	CALL DWORD PTR DS:[418460]	USER32.GetMessageW
0041159E	CALL DWORD PTR DS:[418464]	USER32.TranslateMessage

MessageBox 함수를 호출하는 부분을 찾아가보았다.

Address	Hex dump	Disassembly	Comment
00413D81	3BF4	CMP ESI,ESP	
00413D83	E8 E5D3FFFF	CALL Qnkgd2Fp,0041116D	
00413D88	EB 32	JMP SHORT Qnkgd2Fp,00413D8C	
00413D8A	3BF4	MOV ESI,ESP	
00413D8C	6A 00	PUSH 0	
00413D8E	FF15 34844100	CALL DWORD PTR DS:[418434]	USER32.MessageBeep
00413D94	3BF4	CMP ESI,ESP	
00413D96	E8 D2D3FFFF	CALL Qnkgd2Fp,0041116D	
00413D98	EB 1F	JMP SHORT Qnkgd2Fp,00413D8C	
00413D9D	3BF4	MOV ESI,ESP	
00413D9F	6A 00	PUSH 0	
00413DA1	68 A0624100	PUSH Qnkgd2Fp,004162A0	
00413DA6	68 385A4100	PUSH Qnkgd2Fp,00415A38	UNICODE "SNNHAK1215.DLL"
00413DAB	8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	

SNNHAK1215.DLL 이라는 문자열을 출력하는 구문을 발견했고 키라고 생각하여 인증하니 문제가 클리어되었다.

Key : SNNHAK1215.DLL

- 12 -

[G] Forensic 200P

Subject

Energy

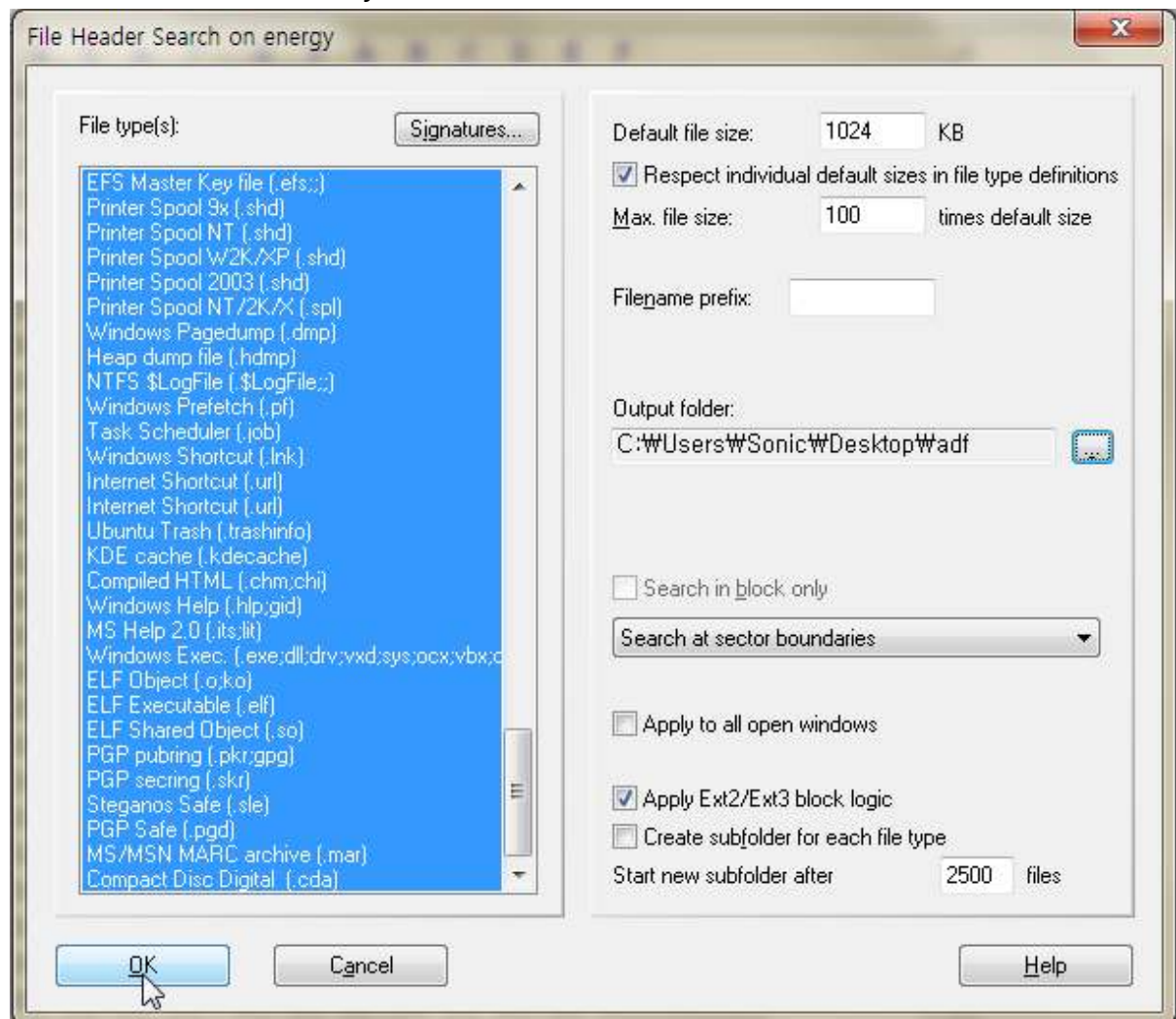
Comment

I'm Your Energy - J.S Park

By. lwmr

File : energy.zip

energy.zip 파일을 압축해제하면 energy 라는 파일이 나온다.
winhex를 이용해 recovery하였다.



다음과 같이 20개의 파일들이 추출되었다.



파일들을 둘러보다가 00008.jpg 에 적혀있는 글귀에 energy라는 글이 있길래 인증하니 답이었다.



Key : Drive Your Energy! Bacchus-D

[N] Forensic 200P

Subject

남친털기

Comment

남자친구의 가방에서 USB를 획득한 영희.

USB의 한 폴더엔 여자의 사진들로 가득했는데...

그 모습을 본 남자친구는 화들짝 놀라며 폴더를 지워버렸다.

영희가 다른 폴더를 뒤질려고 하자 남자친구는 화를 내며 USB를 포맷시켜 버렸다.

화가난 영희는 포맷된 USB를 가져왔는데.....

By. MANO

File : 남친털기

처음에는 winhex로 recovery를 하여 파일들을 추출하니 30개의 이미지가 나왔고, 이미지마다 문자열들이 적혀있었다.

이미지의 이름 순서대로 문자열을 모으니 다음의 값이 나왔다.

c6568ac7a43ec681a6a0a0c2940eca470daf2999917d13b77d1a38ca6ed35bd106628
ebc6828a59f062f8660450e366b

그러나 인증에 실패하였고, 검은 글씨와 흰 글씨 따로 로도 시도해보았으나 실패하였다.

그래서 다른 포렌식 툴을 찾아보다가 리눅스의 Autopsy forensic browser 이라는 툴을 알게 되었다.

Autopsy 툴의 사용법은 다음과 같다.

우선 su root 명령어로 root 권한을 얻은 후 autopsy를 실행시켰다.

(처음엔 autopsy가 깔려 있지 않아 apt-get install autopsy 명령어로 설치를 하였다.)

위와 같이 정상적으로 autopsy가 실행되었고, <http://localhost:9999/autopsy> 주소로 들어가면 아래와 같이 autopsy forensic browser를 사용할 수 있다.

```
root@Xero-vm: /home/sonic
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
root@Xero-vm:/home/sonic# autopsy

=====
Autopsy Forensic Browser
http://www.sleuthkit.org/autopsy/
ver 2.24
=====

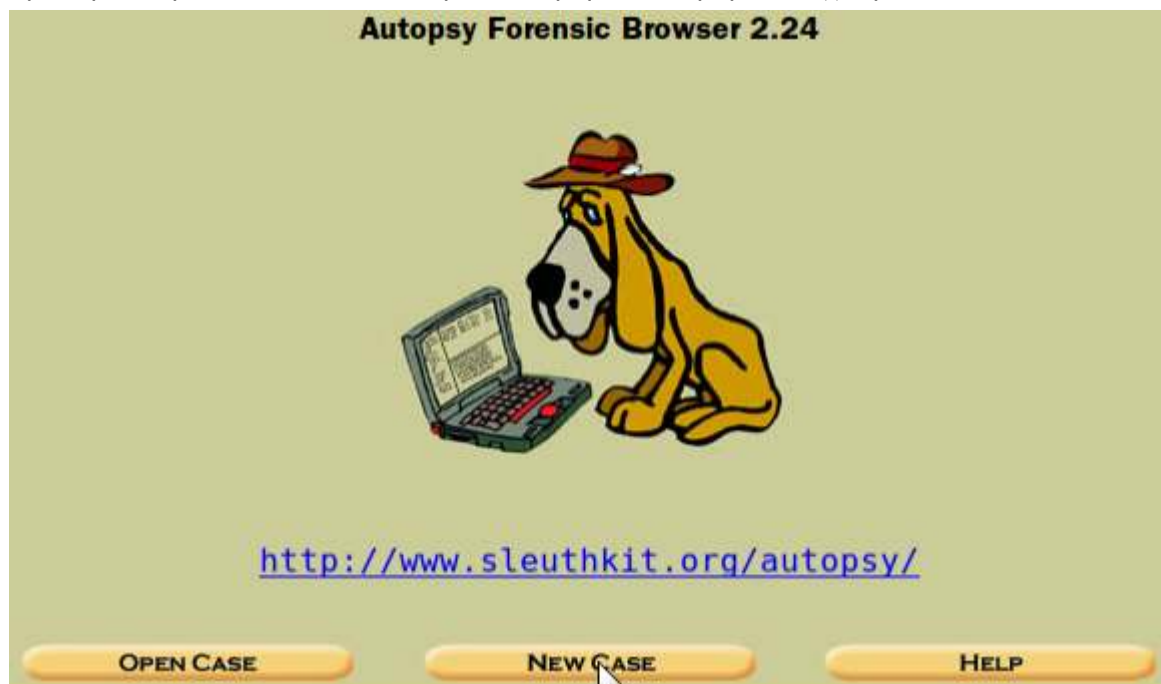
Evidence Locker: /var/lib/autopsy
Start Time: Tue Oct 18 00:01:29 2011
Remote Host: localhost
Local Port: 9999

Open an HTML browser on the remote host and paste this URL in it:

    http://localhost:9999/autopsy

Keep this process running and use <ctrl-c> to exit
█
```

다음과 같이 NEW CASE 로 새로운 케이스를 하나 만들었다.



아래와 같이 Case Name을 입력하고 NEW CASE 버튼을 눌러서 만들었다.

CREATE A NEW CASE

1. **Case Name:** The name of this investigation. It can contain only letters, numbers, and symbols.

Xero

2. **Description:** An optional, one line description of this case.

3. **Investigator Names:** The optional names (with no spaces) of the investigators for this case.

a.		b.	
c.		d.	
e.		f.	
g.		h.	
i.		j.	

NEW CASE CANCEL HELP

다음과 같이 새로운 케이스가 만들어졌고 ADD HOST 버튼으로 HOST를 추가하는 페이지로 넘어갈 수 있다.

Creating Case: Xero

Case directory (/var/lib/autopsy/Xero/) created
Configuration file (/var/lib/autopsy/Xero/case.aut) created

We must now create a host for this case.

ADD HOST

Host Name 을 입력하고 ADD HOST 버튼을 눌러 HOST를 추가하였다.

ADD A NEW HOST

1. **Host Name:** The name of the computer being investigated. It can contain only letters, numbers, and symbols.

2. **Description:** An optional one-line description or note about this computer.

3. **Time zone:** An optional timezone value (i.e. EST5EDT). If not given, it defaults to the local setting. A list of time zones can be found in the help files.

4. **Timeskew Adjustment:** An optional value to describe how many seconds this computer's clock was out of sync. For example, if the computer was 10 seconds fast, then enter -10 to compensate.

5. **Path of Alert Hash Database:** An optional hash database of known bad files.

6. **Path of Ignore Hash Database:** An optional hash database of known good files.

ADD HOST **CANCEL** **HELP**

HOST가 추가되었고, ADD IMAGE 버튼을 눌러 이미지를 추가하는 페이지로 넘어갔다.

No images have been added to this host yet

Select the Add Image File button below to add one

ADD IMAGE FILE **CLOSE HOST**
HELP

FILE ACTIVITY TIME LINES **IMAGE INTEGRITY** **HASH DATABASES**
VIEW NOTES **EVENT SEQUENCER**

ADD IMAGE FILE 버튼을 눌러서 이미지파일을 추가했다.

Location 폼에 대상 파일의 full path를 넣고 Type을 Partition으로 선택하고 NEXT 버튼으로 다음 과정으로 넘어갔다.

(‘남친털기’ 라는 한글 파일 이름을 그대로 넣었더니 에러가 나서 파일 이름을 영어로 변경하였다.)

ADD A NEW IMAGE

1. Location
Enter the full path (starting with /) to the image file.
If the image is split (either raw or EnCase), then enter '*' for the extension.

2. Type
Please select if this image file is for a disk or a single partition.
☐ Disk ☒ Partition

3. Import Method
To analyze the image file, it must be located in the evidence locker. It can be imported from its current location using a symbolic link, by copying it, or by moving it. Note that if a system failure occurs during the move, then the image could become corrupt.
☒ Symlink ☐ Copy ☐ Move

NEXT **CANCEL** **HELP**

Data Integrity를 Calculate로 선택하고 ADD 버튼을 눌러 추가하였다.

Image File Details

Local Name: images/asdf

Data Integrity: An MD5 hash can be used to verify the integrity of the image. (With split images, this hash is for the full image file)

- ☐ Ignore the hash value for this image.
- ☒ Calculate the hash value for this image.
- ☐ Add the following MD5 hash value for this image:
- ☐ Verify hash after importing?

File System Details

Analysis of the image file shows the following partitions:

Partition 1 (Type: fat12)

Mount Point: File System Type:

ADD **CANCEL** **HELP**

OK 버튼을 누르면 이미지가 추가된다.

Calculating MD5 (this could take a while)
Current MD5: 8CB8940E307C1652FDD782930429BF6D
Testing partitions
Linking image(s) into evidence locker
Image file added with ID img1
Volume image (0 to 0 - fat12 - C:) added with ID vol1

OK **ADD IMAGE**

ANALYZE 버튼을 눌러서 분석 해보았다.

Select a volume to analyze or add a new image file.

CASE GALLERY **HOST GALLERY** **HOST MANAGER**

mount	name	fs type	
<input checked="" type="radio"/> C: /	asdf-0-0	fat12	details

ANALYZE **ADD IMAGE FILE** **CLOSE HOST**
HELP

FILE ACTIVITY TIME LINES **IMAGE INTEGRITY** **HASH DATABASES**
VIEW NOTES **EVENT SEQUENCER**

다음과 같이 나타나는데 FILE ANALYSIS 를 눌러 파일 분석을 시도하였다.



다음과 같이 에러없이 분석이 되었고, \$OrphanFiles/ 로 들어가보았다.

Current Directory: C:/

ADD NOTE GENERATE MD5 LIST OF FILES

DEL	Type dir / in	NAME	WRITTEN	ACCESSED	CREATED	SIZE	UID	GID
v / v		\$FAT1	0000-00-00 00:00:00 (UTC)	0000-00-00 00:00:00 (UTC)	0000-00-00 00:00:00 (UTC)	6144	0	0
v / v		\$FAT2	0000-00-00 00:00:00 (UTC)	0000-00-00 00:00:00 (UTC)	0000-00-00 00:00:00 (UTC)	6144	0	0
v / v		\$MBR	0000-00-00 00:00:00 (UTC)	0000-00-00 00:00:00 (UTC)	0000-00-00 00:00:00 (UTC)	512	0	0
d / d		\$OrphanFiles/	0000-00-00 00:00:00 (UTC)	0000-00-00 00:00:00 (UTC)	0000-00-00 00:00:00 (UTC)	0	0	0
r / r		00 0000 (Volume Label Entry)	2011-08-22 21:43:58 (KST)	0000-00-00 00:00:00 (UTC)	0000-00-00 00:00:00 (UTC)	0	0	0

\$OrphanFiles/ 로 들어가면 다음과 같이 이미지들이 이름까지 완벽하게 복원된 것을 볼 수 있다.

Current Directory: C:/ /\$OrphanFiles/

ADD NOTE GENERATE MD5 LIST OF FILES

DEL	Type dir / in	NAME	WRITTEN	ACCESSED	CREATED	SIZE	UID	GID
✓	- / r	ANUARY.JPG	2011-08-22 11:33:50 (KST)	2011-08-22 00:00:00 (KST)	2011-08-22 21:43:26 (KST)	44163	0	0
✓	- / r	ARCH.JPG	2011-08-22 11:34:12 (KST)	2011-08-22 00:00:00 (KST)	2011-08-22 21:43:26 (KST)	82284	0	0
✓	- / r	AY.JPG	2011-08-22 11:37:58 (KST)	2011-08-22 00:00:00 (KST)	2011-08-22 21:43:26 (KST)	191062	0	0
✓	- / r	CTOBER.JPG	2011-08-22 11:36:52 (KST)	2011-08-22 00:00:00 (KST)	2011-08-22 21:43:26 (KST)	96773	0	0
✓	- / r	EBRUARY.JPG	2011-08-22 11:34:02 (KST)	2011-08-22 00:00:00 (KST)	2011-08-22 21:43:26 (KST)	175318	0	0
✓	- / r	ECEMBER.JPG	2011-08-22 11:37:12 (KST)	2011-08-22 00:00:00 (KST)	2011-08-22 21:43:26 (KST)	111964	0	0

다음과 같이 월별, 요일별, 숫자별 등으로 나타나있다.



파일명의 첫 글자는 복구하지 못했지만 계성이 가능한 정도였다.

월별로 이미지의 글자들을 이어 붙이니 다음의 값이 나왔고 인증에 성공했다.

7d1991d1a0da38c773bf2bd1d35a6e99

Key : 7d1991d1a0da38c773bf2bd1d35a6e99

[R] Trivial 200P

Subject

Trivial is fun

Comment

By. Rust

File : trivia_is_fun.png

다음과 같이 TRiViA iS FUN! 이라는 글자가 일정한 패턴으로 이루어져 있다.

TRiViA iS FUN!

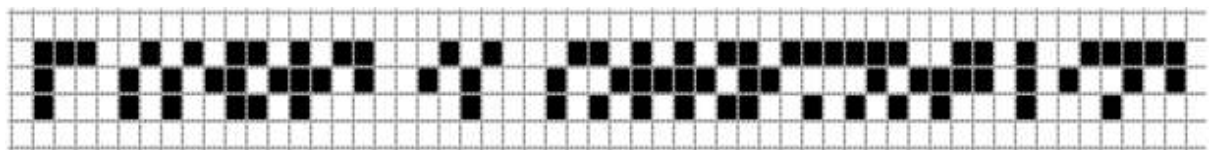
확대하여 글자들의 밑쪽을 보면 hint is blind 라는 글의 글자들이 한 글자씩 나와 있는 것을 볼 수 있다.

힌트가 보이지 않는 것이라고 해서 LSB 워터마크도 시도해보고 명도 채도 값, 레벨 값 변경도 시도해보았는데 아무것도 숨겨져 있지 않았다.

힌트로 점자가 나오자 그제서야 blind를 장님으로 보고 문제를 풀어야 한다는 걸 알았다.

글자들이 일정한 패턴으로 이루어져 있으므로 일정한 패턴을 뽑아보았다.

다음이 그 일정한 패턴이다.



영어 점자 표로 해독하니 password is sorryimnotblind 가 나왔다.

Key : sorryimnotblind

[T] Analysis 200P

Subject

I'm on a local

Comment

난 고약한 해커다!

길을가며 Wifi를 찾다가 취약한 AP를 발견하여 접속을 해보니
어떤 사람이 무슨 작업을 하고 있는듯 한데..?

By. Rust

File : T6133d45d592a75c103d30a3c3f33dcdb.zip

다음과 같이 패킷들이 잡히게 보인다.

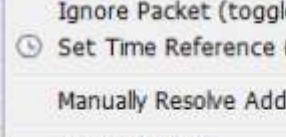
1	0.000000	120.50.133.190	192.168.0.37	TCP	62	avt-pro
2	0.000014	192.168.0.37	120.50.133.190	TCP	60	54811 >
3	0.000076	192.168.0.37	120.50.133.190	TCP	54	[TCP DU
4	0.000179	192.168.0.37	120.50.133.190	TCP	54	[TCP DU
5	0.000431	192.168.0.37	120.50.133.190	TCP	66	54811 >
6	0.000469	192.168.0.37	120.50.133.190	TCP	66	[TCP Re
7	0.000517	192.168.0.37	120.50.133.190	TCP	66	[TCP Re
8	0.006255	120.50.133.190	192.168.0.37	TCP	60	avt-pro
9	0.006278	120.50.133.190	192.168.0.37	TCP	60	[TCP DU

필터를 tcp로 하여 tcp만 뽑아보았다.

Filter: tcp

tcp 패킷들의 stream을 보던 도중 vnc와 함께 많은 패킷들이 오고가는 부분을 보았다.

36	11.776962	192.168.0.37	78.131.193.150	TCP	60	56514 > 40489 [ACK] Seq=1 Ack=1 win=65392 Len=1
37	11.949380	192.168.0.58	192.168.0.29	VNC	70	
38	11.954620	192.168.0.29	192.168.0.58	VNC	60	
39	11.971668	192.168.0.58	192.168.0.29	VNC	60	
40	11.975261	192.168.0.29	192.168.0.58	VNC	97	
41	12.062230	192.168.0.58	192.168.0.29	VNC	74	
42	12.062312	192.168.0.58	192.168.0.29	VNC	114	
43	12.063329	192.168.0.29	192.168.0.58	TCP	60	rfb > danf-ak2 [ACK] Seq=48 Ack=98 win=14600 Len=0
44	12.065050	192.168.0.58	192.168.0.29	VNC	64	
45	12.065876	192.168.0.29	192.168.0.58	VNC	60	
46	12.077985	192.168.0.29	192.168.0.58	VNC	1514	



The screenshot shows the Wireshark interface with a context menu open over a selected packet. The menu options are:

- Mark Packet (toggle)
- Ignore Packet (toggle)
- Set Time Reference (toggle)
- Manually Resolve Address
- Apply as Filter
- Prepare a Filter
- Conversation Filter
- Colorize Conversation
- SCTP
- Follow TCP Stream** (highlighted by the mouse)
- Follow UDP Stream
- Follow SSL Stream
- Copy
- Decode As...

The background packet list shows a selected packet with the following details:

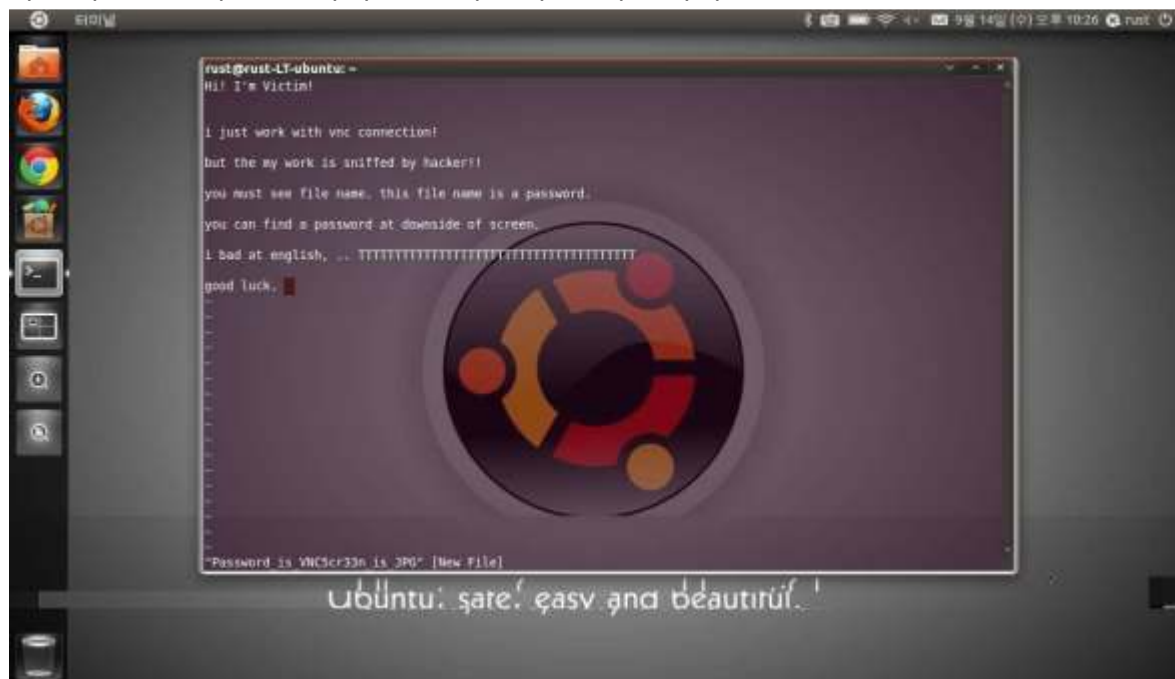
- Seq=48 Ack=...
- ee:66)
- 58.0.58)
- Ack: 98, L...

[illegible]

jpg 파일은 파일 시그니처가 FF D8 FF로 시작하므로 Ctrl+F 로 찾아 앞쪽의 불필요한 헤더들을 제거하였다.

72 01 F6 19 E6 A1 A2 A5 AC 0B 77 8A 27 BF CC B8	r.ö.æ;ç¥¬.wš'zİ.
00 00 00 00 01 05 00 03 20 20 18 00 01 00 FF 00ŷ
FF 00 FF 10 08 00 FB 3C 01 00 00 00 13 72 75 73	ŷ.ŷ...û<.....rus
74 40 72 75 73 74 2D 4C 54 2D 75 62 75 6E 74 75	t@rust-LT-ubuntu
00 00 01 01 20 18 00 01 00 FF 00 FF 00 FF 10 08ŷ.ŷ.ŷ..
00 FB 3C 01 02 43 00 0E 00 00 00 07 00 00 00 08	.û<..C.....
00 00 00 06 00 00 00 05 00 00 00 04 00 00 00 02
00 00 00 01 00 00 00 00 FF FF FF 10 FF FF FF 11ŷŷŷ.ŷŷŷ.
FF FF FF 18 FF FF FF E6 FF FF FF 20 FF FF FF 21	ŷŷŷ.ŷŷŷæŷŷŷ ŷŷŷ!
03 00 00 00 00 00 05 00 03 20 00 00 FF FF 00 00ŷŷ..
00 00 05 00 00 33 00 00 00 07 90 EF 30 FF D8 FF3.....ioŷŷ

다음이 잘린 사진을 이어붙인 바탕화면 사진이다.



Key : VNC5cr33n_is_JPG

[V] Trivial 300P

Subject

ASCII storm

Comment

By. Rust

File : ASCII_STORM.pdf

이 문제는 어려웠지만 재미있게 푼 문제이다.

주어진 pdf를 열어보면 아무 것도 없고 단지 Watch Number 7 :D 라는 글귀만이 있다.

숨겨진 글자를 찾다가 실패했고, 헥스로 뜯어보던 중 수상한 헤더를 발견하였다.

```
3E 3E 0D 0A 73 74 72 65 61 6D 0D 0A 36 3C 23 27 >>..stream..6<#'  
5C 37 50 51 23 46 2B 42 32 71 71 30 65 61 5F 29 \7PQ#F+B2qq0ea_)  
30 48 61 3E 2A 2B 3D 4B 40 24 36 3B 5E 5A 47 33 0Ha>*+=K@$6;^ZG3  
26 3D 66 37 2F 54 4F 2D 70 40 6A 63 2B 4B 3F 56 &=f7/TO~p@jc+K?V  
3C 37 63 30 35 72 3D 4F 39 31 44 3C 63 31 2E 45 <7c05r=O91D<c1.E  
70 37 33 2B 6B 39 2B 31 64 51 31 4F 35 57 56 59 p73+k9+1dQ1O5WVY  
5E 2D 6E 6F 37 2C 46 26 6B 25 46 3B 44 55 66 74 ^~no7,F&k%F;DUft  
41 39 31 25 50 32 30 3A 23 6A 30 2F 2E 21 46 2E A91%P20:#j0/.!F.  
53 29 3F 44 34 46 54 49 43 2B 42 33 28 75 37 38 S)?D4FTIC+B3(u78  
73 7E 3E 0D 0A 65 6E 64 73 74 72 65 61 6D 0D 0A s~>..endstream..
```

..endstream.. 앞의 ~>는 base85 암호화이다.

그러나 앞의 <~ 가 없어서 base85가 아니라 여기고 여러 시도를 하던 중 ..stream.. 뒤쪽인 6<#W7PQ#F+B2qq0ea_)0Ha>*+=K@\$6;^ZG3&=f7/TO~p@jc+K?V<7c05r=O91D<c1.Ep73+k9+1dQ1O5WVY^~no7,F&k%F;DUftA91%P20:#j0/.!F.S)?D4FTIC+B3(u78s~> 전체를 base85 디코드 해 보았다.

no7,F&k%F;DUftA91%P20:#j0/.!F.S)?D4FTIC+B3(u78s~> 전체를 base85 디코드 해 보았다.

디코드 하니 다음의 값이 나왔다.

BT /F1 8 Tf 10 10 Td (<~BQS?83WN-rAnc'm2_K5b/p(fKFDI2F/n8g:04AsE@:Nt(0fLsV2)R3G1dsAk5t"/(0f_*H3(<~>) Tj ET

이번에는 <~ ~> 모두 있어 base85로 바로 해독하니 다음의 주소가 나왔다.

<http://cfile7.uf.tistory.com/attach/175B55424E8CADCE19528F>

위의 주소로 들어가 txt 파일 하나를 다운받았다.

열어보니 헥스 같아 보이는 값들이 나와있었다.

아스키를 16진수로 바꿀까, 16진수를 아스키로 바꿀까 하다가 그냥 아스키를 16진수로 바꿔서 헥스 에디터에 집어넣기로 했다.

다음이 파이썬으로 코딩한 소스이다.

```
# -*- coding: cp949 -*-
```

```
f=file('C:/Users/Sonic/Desktop/45c!!t0l-leX.txt') #파일 열기
```

```
sProblem=f.read() #파일에서 전체 읽음
```

```
f.close() #파일 닫기
```

```
lAnswer=[] #배열 정의
```

```
sSplit=sProblem.split() #공백단위로 나눔
```

```
for i in range(len(sSplit)): #len함수로 배열의 갯수를 구해 for문을 돌림
```

```
if(len(sSplit[i])==1): #헥스와 아스키를 구분하는것이므로 길이가 1이면 아스키일것이다
```

```
lAnswer.insert(i,hex(ord(sSplit[i]))[2:]) #아스키를 10진수로바꿔 헥스로 바꾼후 슬라이싱을 통해 3자리부터 끊는다 (0x를 없애기위해)
```

```
else: #만약 헥스라면
```

```
lAnswer.insert(i,sSplit[i]) #그대로 추가
```

```
f=file('C:/Users/Sonic/Desktop/Answer.txt','w') #파일 열기
```

```
f.write("".join(lAnswer)) #배열 전체 출력
```

```
f.close() #파일 닫기
```

위의 프로그램을 실행시켜 헥스 값을 얻었고 헥스 에디터에 넣었다.

```
43 57 53 08 49 B8 00 00 78 9C ED 7D 07 5C 53 C9 CWS.I,...xœi}.\SÉ
F6 FF A4 41 42 00 91 5E 15 91 AE 40 A8 36 4A 28 öÿ*AB.'^.'@@"6J(
D2 09 08 88 20 08 04 08 10 09 04 43 68 56 76 AD Ò..^ .....ChVv.
6B 17 29 22 2A 58 96 15 7B 05 DB 2A 6B 17 75 5D k.)"*X-.{.Û*k.u]
5D 75 C5 DE 7B 77 ED 85 FF BD 73 6F 92 9B 10 44 ]uÄP{wi...ÿ*so'>.D
```

파일 시그니처가 43 57 53 이므로 확장자를 swf로 바꾸고 재생해 보았다.
다음과 같이 정상적으로 재생되었고 인증에 성공했다.



Key : Do_U_L0V3_C0de?

[W] Analysis 300P

Subject

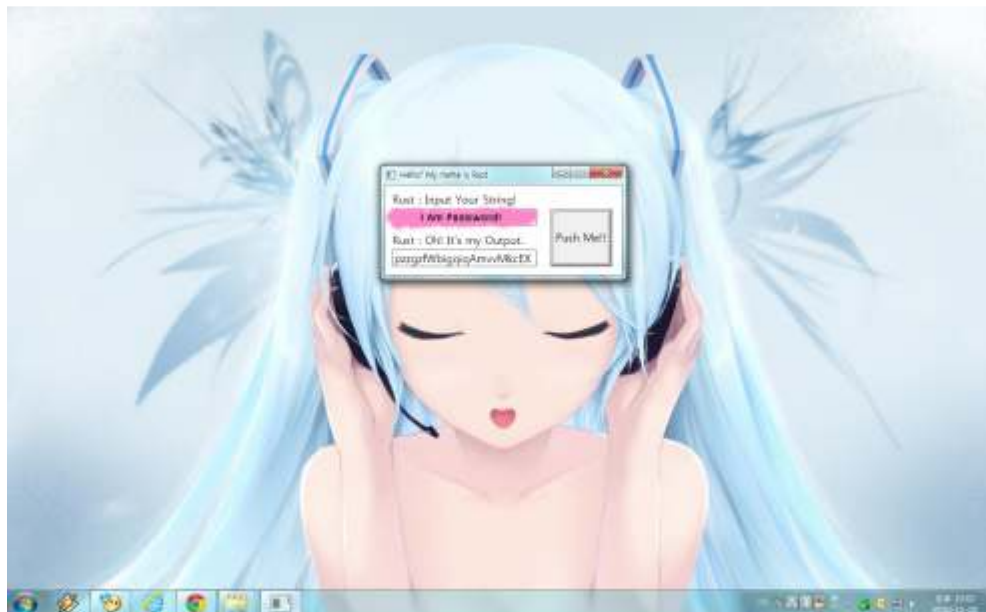
Time is gold

Comment

By. Rust

File : ReverseMe.zip

압축을 해제하면 ReverseMe.exe 파일과 함께 다음의 사진 한 장이 주어진다.



여타 리버싱 문제와 같이 문자열을 입력하면 내부의 소스에 따라 처리되어 암호화 된 문장이 나온다.

우리가 할 일은 입력한 문자열을 리버싱을 통해 찾는 것이다.

올리디버거로 다음과 같이 GetWindowTextA() 함수에 브레이크 포인트를 걸고 버튼을 눌러 암호화 시작 부분을 분석했다.

우선 문자열의 길이가 0x1A(26) 글자 이상인지 확인한다.

그리고 다음으로 문자열이 알파벳으로만 이루어 졌는가를 확인한다.

00061140	FF15 FC200600	CALL DWORD PTR DS:[<&USER32.GetWindowTextA]	GetWindowTextA
00061153	83F8 1A	CMP EAX,1A	
00061156	7C 29	JL SHORT ReverseM,00061181	
00061158	A1 78330600	MOV EAX,DWORD PTR DS:[63378]	
0006115D	6A 00	PUSH 0	
0006115F	68 8C210600	PUSH ReverseM,0006218C	Style = MB_OK MB_APPLMODAL
00061164	68 94210600	PUSH ReverseM,00062194	Title = "ERROR"
00061169	50	PUSH EAX	Text = "Don't Edit This!"
0006116A	FF15 00210600	CALL DWORD PTR DS:[<&USER32.MessageBoxA]	hOwner => 0012089E ('Hello? My na
00061170	8B00 74330600	MOV ECX,DWORD PTR DS:[63374]	MessageBoxA
00061176	68 A8210600	PUSH ReverseM,000621A8	ASCII "X_X!"
0006117B	51	PUSH ECX	
0006117C	E9 42020000	JMP ReverseM,000613C3	
00061181	807D E0 00	CMP BYTE PTR SS:[EBP-20],0	
00061185	74 20	JE SHORT ReverseM,000611A7	
00061187	8B3D 60200600	MOV EDI,DWORD PTR DS:[<&MSVCR100.iswalp	MSVCR100.iswalpha
0006118D	8D75 E0	LEA ESI,DWORD PTR SS:[EBP-20]	
00061190	66:0FBF16	MOVSB DX,BYTE PTR DS:[ESI]	
00061194	0FB7C2	MOVZX EAX,DX	
00061197	50	PUSH EAX	
00061198	FFD7	CALL EDI	
0006119A	83C4 04	ADD ESP,4	
0006119D	85C0	TEST EAX,EAX	
0006119F	74 2F	JE SHORT ReverseM,000611D0	
000611A1	46	INC ESI	
000611A2	803E 00	CMP BYTE PTR DS:[ESI],0	
000611A5	75 E9	JNZ SHORT ReverseM,00061190	
000611A7	33DB	XOR EBX,EBX	
000611A9	385D E0	CMP BYTE PTR SS:[EBP-20],BL	

다음은 첫 번째 암호화 루틴이다.

000611B0	0FBF541D E0	MOVSB EDX,BYTE PTR SS:[EBP+EBX-20]	
000611B5	8D441A 01	LEA EAX,DWORD PTR DS:[EDX+EBX+1]	
000611B9	50	PUSH EAX	
000611BA	FF15 6C200600	CALL DWORD PTR DS:[<&MSVCR100.isalpha>]	isalpha
000611C0	83C4 04	ADD ESP,4	
000611C3	85C0	TEST EAX,EAX	
000611C5	74 1A	JE SHORT ReverseM,000611E1	
000611C7	8D4B 01	LEA ECX,DWORD PTR DS:[EBX+1]	
000611CA	004C1D E0	ADD BYTE PTR SS:[EBP+EBX-20],CL	
000611CE	EB 18	JMP SHORT ReverseM,000611E8	
000611D0	8B0D 74330600	MOV ECX,DWORD PTR DS:[63374]	
000611D6	68 80210600	PUSH ReverseM,000621B0	ASCII "Use Only Alphabet!"
000611DB	51	PUSH ECX	
000611DC	E9 E2010000	JMP ReverseM,000613C3	
000611E1	8D53 E7	LEA EDX,DWORD PTR DS:[EBX-19]	
000611E4	00541D E0	ADD BYTE PTR SS:[EBP+EBX-20],DL	
000611E8	43	INC EBX	
000611E9	807C1D E0 00	CMP BYTE PTR SS:[EBP+EBX-20],0	
000611EE	75 C0	JNZ SHORT ReverseM,000611B0	

0x000611B0 - EDX에 문자열을 한 글자씩 가져옴
 0x000611B5 - EAX에 가져온 문자열 + EBX + 1을 함 (EBX는 1씩 증가됨)
 0x000611C5 - 더한 값이 올바른 알파벳인지 확인 함
 0x000611C7 - 알파벳이 맞다면 메모리에 씀
 0x000611E1 - 알파벳이 아니라면 가져온 문자열 + EBX - 0x19를 메모리에 씀

이를 복호화 가능하도록 다음과 같이 코딩하였다.

```
for (i=0; i<nLen; i++) {
    if ((szInput[i] >= 'a') && (szInput[i] <= 'z')) {
        if ((szInput[i] - i) <= 'a') {
            szInput[i] -= i - 0x19;
        } else {
            szInput[i] -= i + 1;
        }
    } else {
        if ((szInput[i] - i) <= 'A') {
            szInput[i] -= i - 0x19;
        } else {
            szInput[i] -= i + 1;
        }
    }
}
```

그리고 다음과 같이 암호화 첫 단계를 거친 문자열을 뒤집는다.

00061210	>	8A5C05 E0	MOV BL, BYTE PTR SS:[EBP+EAX-20]
00061214	.	8A540D E0	MOV DL, BYTE PTR SS:[EBP+ECX-20]
00061218	.	885C0D E0	MOV BYTE PTR SS:[EBP+ECX-20], BL
0006121C	.	885405 E0	MOV BYTE PTR SS:[EBP+EAX-20], DL
00061220	.	41	INC ECX
00061221	.	48	DEC EAX
00061222	.	3BC8	CMP ECX, EAX
00061224	^	7C EA	JL SHORT ReverseM, 00061210

그 아래부터 0x00061370 까지는 현재 시간을 바탕으로 시간 테이블을 생성한다.

Address	Hex dump	ASCII
0015F788	02 00 00 00 00 00 00 00 01 00 00 00 01 00 00 00F.....F....
0015F798	01 00 00 00 00 00 00 00 01 00 00 00 07 00 00 00F.....F.....
0015F7A8	02 00 00 00 01 00 00 00 00 00 00 00 02 00 00 00F.....F.....

위의 테이블을 4개 단위로 읽어보면 2 0 1 1 1 0 1 7 2 1 0 2 이다.

이 당시의 시간은 다음과 같다.



즉 2 0 1 1 1 0 1 7 2 1 0 2는 2011년 10월 17일 21시 2분이므로 현재의 시간을 테이블로 생성한다는 것을 알 수 있다.

_time64() 함수로 현재 시간을 구하고, _localtime64()으로 변환해서 위와 같이 정수형 배열에 넣는다.

다음으로 0x00061370 ~ 0x000613B6 부분은 두 번째 암호화 부분이다.

00061370	>	83FF 0C	CMP EDI,0C	
00061373	↘	75 02	JNZ SHORT ReverseM,00061377	
00061375	.	33FF	XOR EDI,EDI	
00061377	>	8B44BD B0	MOV EAX,DWORD PTR SS:[EBP+EDI*4-50]	
0006137B	.	83F8 1A	CMP EAX,1A	
0006137E	↘	7E 0C	JLE SHORT ReverseM,0006138C	
00061380	>	83E8 1A	SUB EAX,1A	
00061383	.	8944BD B0	MOV DWORD PTR SS:[EBP+EDI*4-50],EAX	
00061387	.	83F8 1A	CMP EAX,1A	
0006138A	^	7F F4	JG SHORT ReverseM,00061380	
0006138C	>	0FBED0E	MOVSX ECX,BYTE PTR DS:[ESI]	
0006138F	.	034CBD B0	ADD ECX,DWORD PTR SS:[EBP+EDI*4-50]	
00061393	.	51	PUSH ECX	
00061394	.	FF15 6C20060	CALL DWORD PTR DS:[<&MSVCR100.isalpha>]	c isalpha
0006139A	.	83C4 04	ADD ESP,4	
0006139D	.	85C0	TEST EAX,EAX	
0006139F	↘	74 08	JE SHORT ReverseM,000613A9	
000613A1	.	8A54BD B0	MOV DL,BYTE PTR SS:[EBP+EDI*4-50]	
000613A5	.	0016	ADD BYTE PTR DS:[ESI],DL	
000613A7	↘	EB 08	JMP SHORT ReverseM,000613B1	
000613A9	>	8A44BD B0	MOV AL,BYTE PTR SS:[EBP+EDI*4-50]	
000613AD	.	2C 1A	SUB AL,1A	
000613AF	.	0006	ADD BYTE PTR DS:[ESI],AL	
000613B1	>	46	INC ESI	
000613B2	.	47	INC EDI	
000613B3	.	803E 00	CMP BYTE PTR DS:[ESI],0	
000613B6	^	75 B8	JNZ SHORT ReverseM,00061370	

0x0006138C - 문자열의 한 글자씩 가져옴.

0x0006138F - 가져온 문자열에 시간 배열의 숫자를 더함.

0x0006139F - 결과가 알파벳이 맞는지 확인함.

0x000613A5 - 알파벳이 맞을 경우 메모리에 씀.

0x000613A9 - 알파벳이 아닌 경우 시간 배열의 숫자 - 0x1A 만큼을 더해서 메모리에 씀.

이를 복호화 가능하도록 다음과 같이 프로그래밍하였다.

```
j = 0;
for (i=0; i<nLen; i++) {
    if (j >= 0xC) j = 0;

    if ((szInput[i] >= 'a') && (szInput[i] <= 'z')) {
        if ((szInput[i] - nTable[j]) <= 'a') {
            szInput[i] -= nTable[j] - 0x1A;
        } else {
            szInput[i] -= nTable[j];
        }
    } else {
        if ((szInput[i] - nTable[j]) <= 'A') {
            szInput[i] -= nTable[j] - 0x1A;
        } else {
            szInput[i] -= nTable[j];
        }
    }
    j++;
}
```

문제의 사진과 같이 암호화된 문자열에 pzzgzfWbigqiqAmvvMkcEX를 넣고 날짜를 2010-11-28 22:02 로 지정하면 다음과 같이 복호화 된다.

UAreGoodReverseEngineer

Key : UAreGoodReverseEngineer

[Y] Trivial 300P

Subject

천국을 보여줄게

Comment

정말 천사같은 여자야

By. MinAmi33

File : MinAmi33.zip

압축을 해제하면 다음의 이미지 하나가 나타난다.



헥스 에디터로 열어서 보던 도중 안에 다음과 같이 압축파일이 숨어있는 것을 발견했다.

```
4D C0 5C 18 D0 BF FF D9 50 4B 03 04 14 00 00 00 MA\..Đ;YÙPK....  
08 00 D1 92 43 3F 20 14 B3 0A 50 04 00 00 8C 06 ..Ñ'C? .'.P...Æ.
```

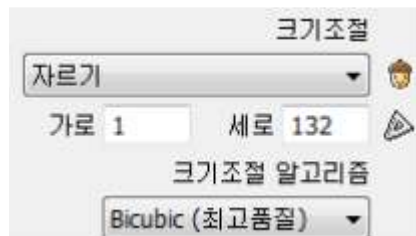
압축을 풀니 컬러 바코드 같아 보이는 이미지들이 132개가 나타났고, 크기 또한 132x132였다.

컬러 바코드를 검색하여 계속 찾았으나 실패하였다.

옆으로도 이어보고 흑백으로 바꿔 겹쳐도 보다가 1x132로 이미지를 잘라 옆으로 이어붙여서 132x132의 이미지를 새롭게 만들어보았다.

일괄 편집을 하기 위해 포토스케이프 툴을 이용했다.

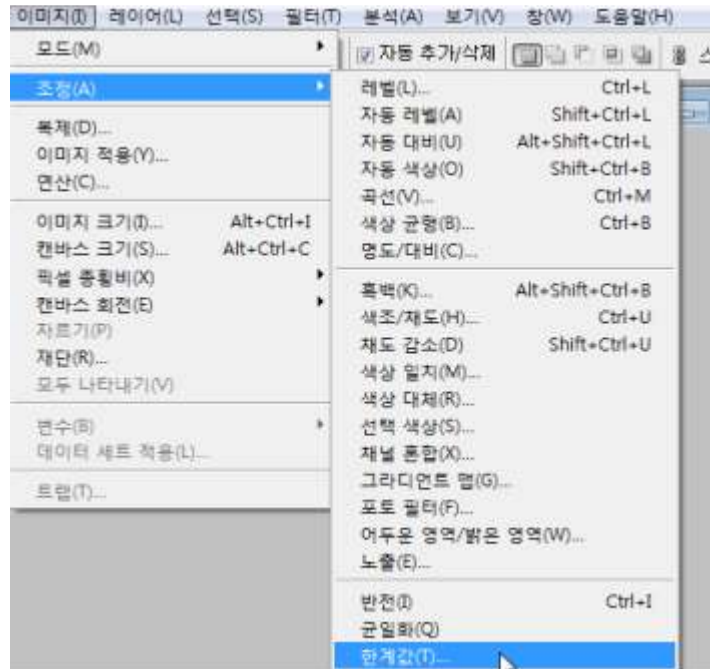
다음과 같이 자르기로 1x132를 지정하여 잘랐다.



그리고 이어붙이기를 하니 다음과 같은 이미지가 나왔고 QR코드라는 것을 알 수 있었다.



포토샵의 이미지의 조정의 한계값 기능으로 처리해보기로 했다.



다음과 같이 한계값을 223 정도로 설정해서 QR코드가 인식 될 정도로 만들었다.



그렇게 만든 다음의 QR코드를 아이폰의 QR스캐너 어플로 스캔했다.
QR코드를 찍자 아래의 주소가 나왔다.

<http://m.site.naver.com/01Yzy>

위의 주소로 들어가자 다음의 글귀가 있었고 인증에 성공했다.

Key is : C_ProgramFiles_Pruna_Incoming_EBS

Key : C_ProgramFiles_Pruna_Incoming_EBS

후기

6위를 하여 입상을 하게 되었다.

여러 분야들의 문제가 나왔고 정말 재미있게 풀었다.

특히 네이투 암호화 문제는 최근 네이트 해킹 유출 이슈를 다뤄서 재미있었다.

시스템 분야가 매우 약하다는 사실을 알게 되었고, 앞으로는 더욱 열심히 공부해야겠다고 느꼈다.

다음은 대회 종료 당시의 랭킹이다.

2200 점으로 6등을 하였다.

Rank	Nick Name	Challenge Point	Last Auth Time
1	인간남케홀마	5400	[16] 04:26:54
2	pwn3r	3140	[16] 10:00:57
3	SecuRex0	2930	[16] 09:24:27
4	LulzSec	2810	[16] 03:42:19
5	extr	2200	[16] 09:41:27
6	Xero	2200	[16] 09:44:48
7	Hello	2100	[16] 07:41:52
8	두루물술	2100	[16] 10:30:04
9	nagi	1910	[16] 05:07:46
10	fuck	1700	[15] 23:48:35
11	Gogil	1510	[16] 04:42:44
12	pepper	1400	[16] 03:06:47
13	B10SM4N	1200	[16] 10:22:32
14	나는_자연인이다	700	[16] 01:37:34
15	ffaass	500	[16] 01:42:00
16	NellP	400	[16] 07:23:08
17	SecurityFirst	300	[15] 15:42:36
18	Loup_	20	[15] 09:24:29
19	freedom	20	[15] 12:23:29

다음은 종료 당시의 챌린지 보드판 현황이다.

[A] Web - 100 P OTHER LulzSec	[B] Crypto - 100 P OTHER nagi	[C] Trivial - 100 P CLEAR 인간남제출마	[D] Trivial - 100 P CLEAR SecuRex0	[E] Analysis - 100 P CLEAR Gogil
[F] Crypto - 200 P CLEAR SecuRex0	[G] Forensic - 200 P CLEAR 인간남제출마	[H] Trivial - 400 P H	-	[J] Crypto - 300 P OTHER 인간남제출마
[K] Analysis - 300 P OTHER 인간남제출마	[L] Forensic - 300 P OTHER 인간남제출마	-	[N] Forensic - 200 P CLEAR 인간남제출마	[O] Forensic - 300 P OTHER 인간남제출마
-	[Q] Analysis - 400 P Q	[R] Trivial - 200 P CLEAR 인간남제출마	[S] Web - 300 P OTHER 인간남제출마	[T] Analysis - 200 P CLEAR 인간남제출마
[U] Forensic - 300 P OTHER 인간남제출마	[V] Trivial - 300 P CLEAR 인간남제출마	[W] Analysis - 300 P CLEAR 인간남제출마	-	[Y] Trivial - 300 P CLEAR 인간남제출마
[Z] System - 400 P OTHER 인간남제출마	-	[Alt] Analysis - 400 P Alt	[Del] Analysis - 500 P Del	[Win] System - 400 P OTHER Anonymous

다음번에는 더욱 많은 문제를 풀어서 보드판을 파란색으로 가득 채우고 싶다.