Inventory

Contents

1 Cryptograph

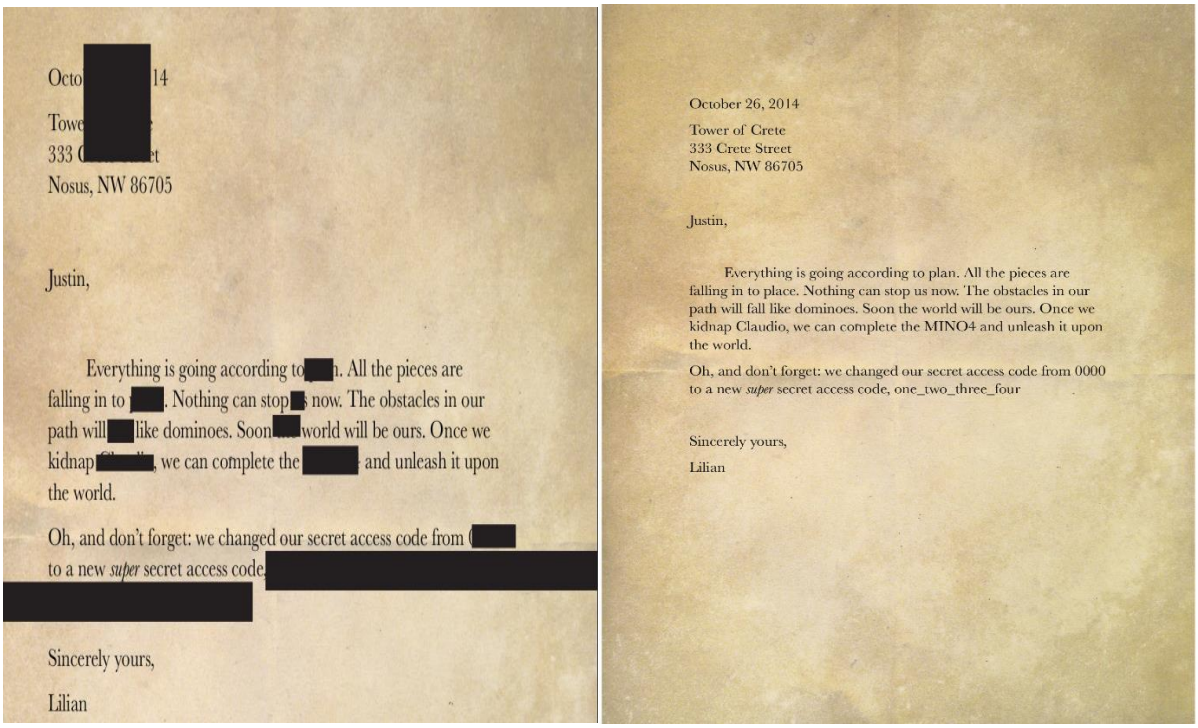
1.1 Packet Analysis

1.1.1 Description: File formats/sub-file (file in image)

Hide information in different kinds of files, maybe contains some obscure file format they've never heard of, and unpacking the entire chain eventually gives them the flag.

1.1.2 Example 1: Executable .pdf file

Add a .zip sub-file in an image file, hide flag in the .zip sub-file /Hide words in file



October 26, 2014

Tower of Crete
333 Crete Street
Nosus, NW 86705

Justin,

Everything is going according to plan. All the pieces are falling in to place. Nothing can stop us now. The obstacles in our path will fall like dominoes. Soon the world will be ours. Once we kidnap Claudio, we can complete the MINO4 and unleash it upon the world.

Oh, and don't forget: we changed our secret access code from 0000 to a new *super* secret access code, one_two_three_four

Sincerely yours,

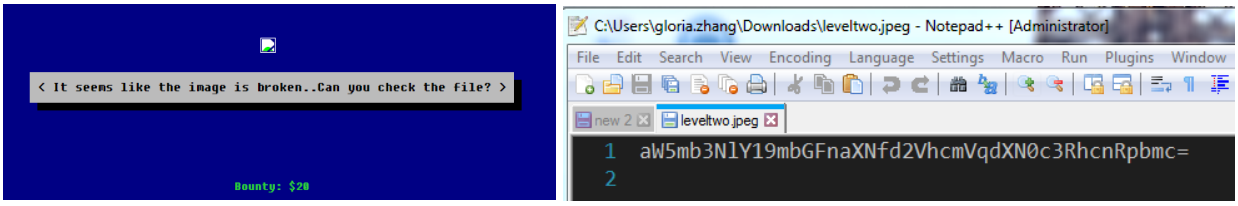
Lilian

(before/ after extract--- http://www.extractpdf.com/)

1.1.3 Example 2: Magic image

Fake information identity, Hide information in picture:

Question Link: http://ctf.infosecinstitute.com/leveltwo.php



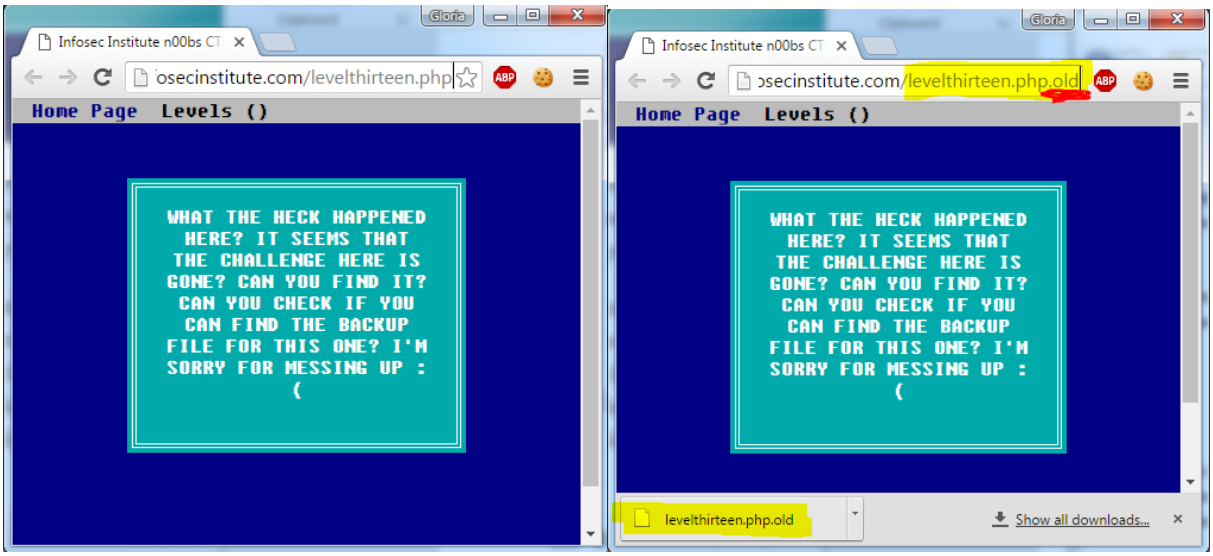
Solution: Save the "broken" picture, and open with notepad, then you'll find the base64 code, convert base64 to ASCII use the online auto convert,you will get the flag.

(aW5mb3NlY19mbGFnaXNfd2VhcmVqdXN0c3RhcnRpbmc=) → (infosec_flagis_wearejuststarting)

1.1.4 Example 3: Related examples

Question description link: http://ctf.infosecinstitute.com/levelthirteen.php

Figure 1: the original webpage. Figure 2: the backup file



For the download backup file, there should be some hints in the file, finally we find the following hints in the file:

```
78      <h1>
79      What the heck happened here? It seems that the challenge here is gone?
80      Can you find it? Can you check if you can find the backup file for this one?
81      I'm sorry for messing up :(
82
83      </h1>
84      <?php
85
86      /* <img src="img/clippy1.jpg" class="imahe" /> <br /> <br />
87
88      <p>Do you want to download this mysterious file?</p>
89
90      <a href="misc/imadecoy">
91        <button class="btn">Yes</button>
92      </a>
93
94      <a href="index.php">
95        <button class="btn">No</button>
96      </a>
97      */
```

Then download the “mysterious file” following the herf, which indicates a hyperlink/location(http://ctf.infosecinstitute.com/misc/imadecoy) of the target file.

Download the file “imadecoy” and analysis it use Wireshark HTTP object analysis module, in Wireshark by clicking File –> Export Objects –> HTTP. Doing so will give you a listing, Save all the files and find the flag.

1.1.5 Tricky& Suggestion

1.1.5.1 Tricky analysis:

i. A very common convention to backup files in Linux is just to add “.old” to the end of the file.
ii. We should never trust what is provided to us. In this case, they gave us a file ending in .jpeg that was actually just a plain text file.
iii. Knowing the type of file can save a lot of time later trying to figure it out

1.1.5.2 Question design suggestion:

1. Give the player a webpage
2. Hide some hints in the “.PHP” file which could help them find the other new file.
3. For the new file, it can be designed has a fake expand name.
4. It could be extracted or read use notepad++/Wireshark, but not the program corresponding with their original file type.
5. Hide encode flag info in the new file.
6. The flag info can be encoded by different cipher(reference: 1.2 Encode/Decode).

1.2 Encode/Decode

1.2.1 Description: Base64/Base32/Hex/Octal/Binary/Morse Code/ASCII convert

Give information in different numeral system, and could be transfer to text(ASCII) in some methods. These methods can be one step transferred or must be transferred for several steps in a specific order.

Binary code:

A bit string, interpreted as a binary number, can be translated into a decimal number. For example, the lower case a, if represented by the bit string 01100001 (as it is in the standard ASCII code), can also be represented as the decimal number 97.

Hexadecimal:

most often the symbols 0–9 to represent values zero to nine, and A, B, C, D, E, F (or alternatively a, b, c, d, e, f) to represent values ten to fifteen

Base64:

Base64 implementation uses A–Z, a–z, and 0–9 for the first 62 values and / for 63 (value table below). Usually end with “=”.

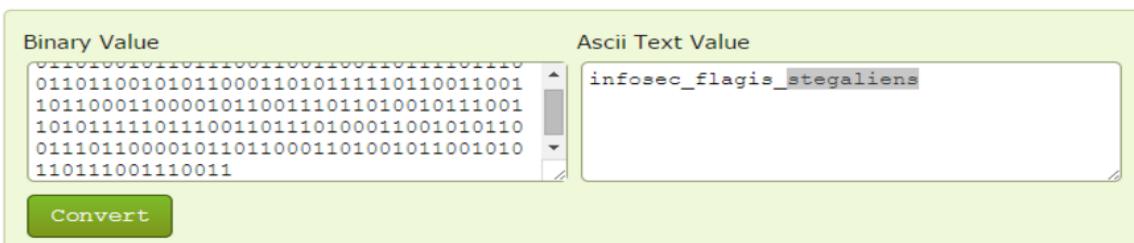
Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

1.2.2 Example 1: Binary Code

Question Description:

https://travisroyer.wordpress.com/2015/03/14/n00bs-ctf-labs-level-5-solution/

First extract this jpg file, you will get a long binary output, then convert binary to ASCII.



1.2.3 Tricky& Suggestion

1.2.3.1 Problem Design Suggestion

i. Combine different number based code as many as possible.

ii. Make the decode process an order, they must follow a exact order t decode, or they won't get the correct flag.

1.3 Letter expression:

1.3.1 Description: Cipher/ Letter frequency/Leet code

express the English words use different symbols, required the attackers to find the rules about the new expression methods, then translate the information to English.

1.3.2 Example 1: Caesar Cipher

ROT 13/ ROT 47/Switch, +/- position, you could get a cipher, where you can guess some of the plaintext, and use that to figure out the key and decrypt the remainder of the message.

1.3.2.1 For example, the input text is: "Gb or be abg gb or, Gung vf gur dhrfgvba", For different ROT-X, it will give the following decode text:

ROT9	Pk xa kn jkp pk xa, Pdwp eo pda mqaopekj
ROT10	Ql yb lo klq ql yb, Qexq fp qeb nrbpqflk
ROT11	Rm zc mp lmr rm zc, Rfyr gq rfc oscqrgml
ROT12	Sn ad nq mns sn ad, Sgzs hr sgd ptdrshnm
ROT13	To be or not to be, That is the question
ROT14	Up cf ps opu up cf, Uibu jt uif rvftujpo
ROT15	Vq dg qt pqv vq dg, Vjcv ku vjg swguvkqp
ROT16	Wr eh ru qrw wr eh, Wkdw lv wkh txhvwlrq
ROT17	Xs fi sv rsx xs fi, Xlex mw xli uyiwxmsr
ROT18	Yt gj tw sty yt gj, Ymfy nx ymj vzjxynts
ROT19	Zu hk ux tuz zu hk, Zngz oy znk wakyzout

1.3.3 Example 2: Leet Code

A password database with unsalted password hashes,

such as 'Leet l33t' alphabet: (https://qntm.org/l33t)

A	/-\ or /\ or 4 or @
B	|3 or 8 or |o
C	(or < or **K** or **S**
D	|) or o| or |> or <|
E	3
F	|= or ph
G	(or 9 or 6
H	|-| or]-[or }-{ or (-) or)-(or #
I	l or 1 or | or ! or][
J	_|
K	|< or /< or \< or |{
L	|_ or | or 1

M	|\/| or /\/\ or |'|'| or (\/) or /\\ or /| or /v\
N	|\\\| or /\/ or |\\\| or /|/
O	o or () or [] or {}
P	|2 or |D
Q	(,) or kw
R	|2 or |Z or |?
S	5 or $
T	+ or][' or 7
U	|_|
V	|/ or |\ or \/ or /
W	\/\/ or ||| or |/|/ or ||/ or |\/ or \^/ or //
X	>< or }{
Y	`/ or '/ or j
Z	2 or (\)

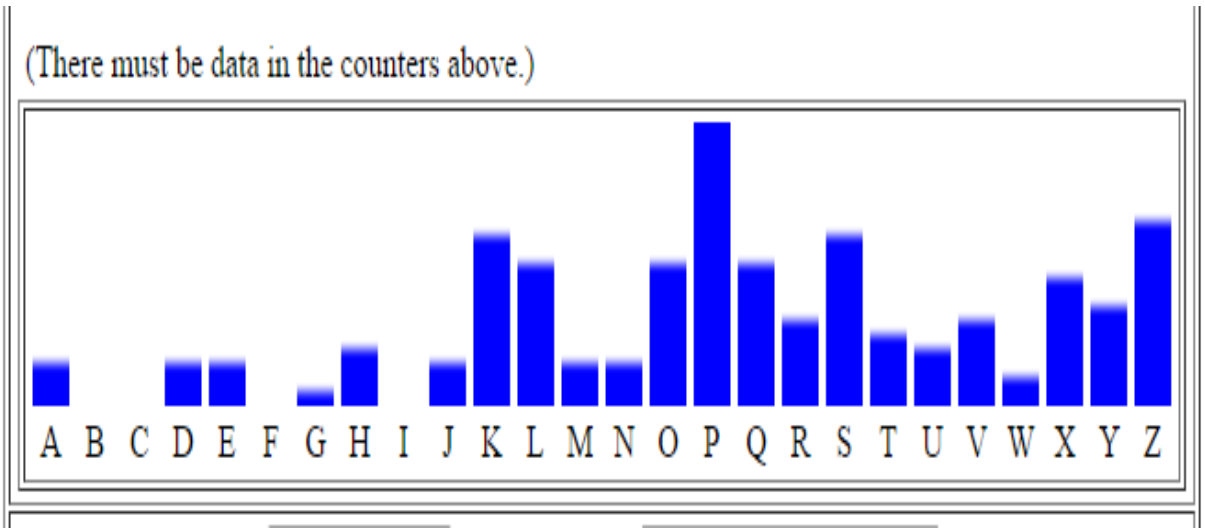
1.3.3.1 Short example: I33t alphabet Table

P@55w0rd_!5_f1@9 => password_is_flag

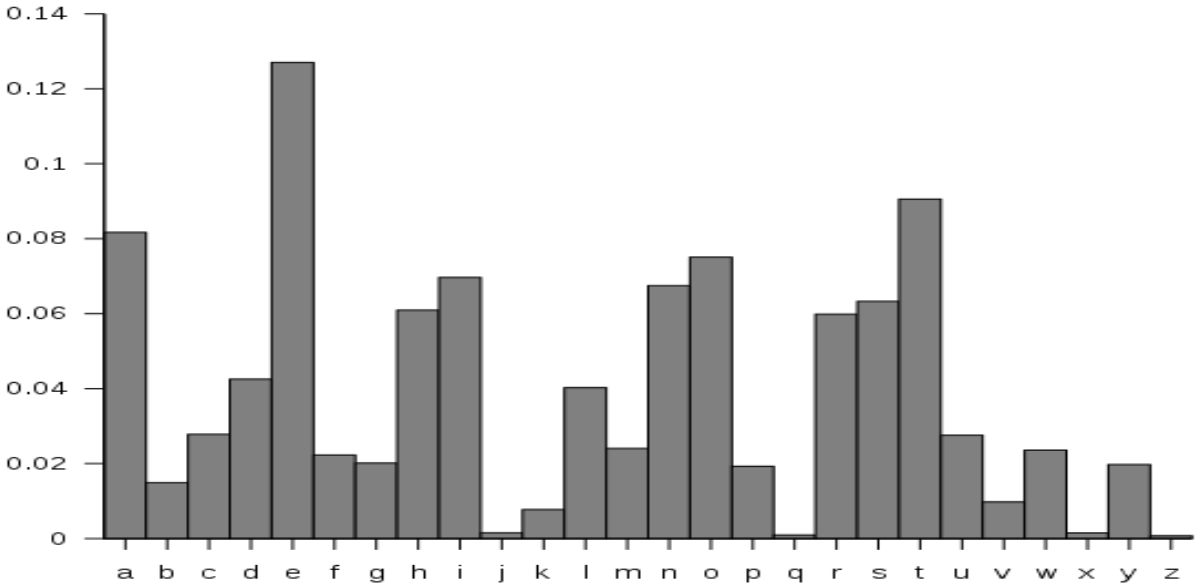
1.3.4 Example 3: Letter Frequency

Question description link:

https://picoctf.com/api/autogen/serve/encrypted.txt?static=false&pid=0cffe6fd67e39c91501cb9d843984cb2



(encode letter frequency ↑) (letter frequency of real English words ↓)



English has limited two letter words/ three letter words. Which makes the crypto easier. Especially some doubly repeated letter and some suffix and prefix and one letter word('a').

We notice that above, we needed to use some information about the frequency of bigrams, two-letter words, and doubly repeated letters in English text in order to confirm certain guesses. The most common bigrams in English text, in order from most to least common, are as follows:

```
th er on an re he in ed nd ha at en es of
```

The most frequent 2-letter words are:

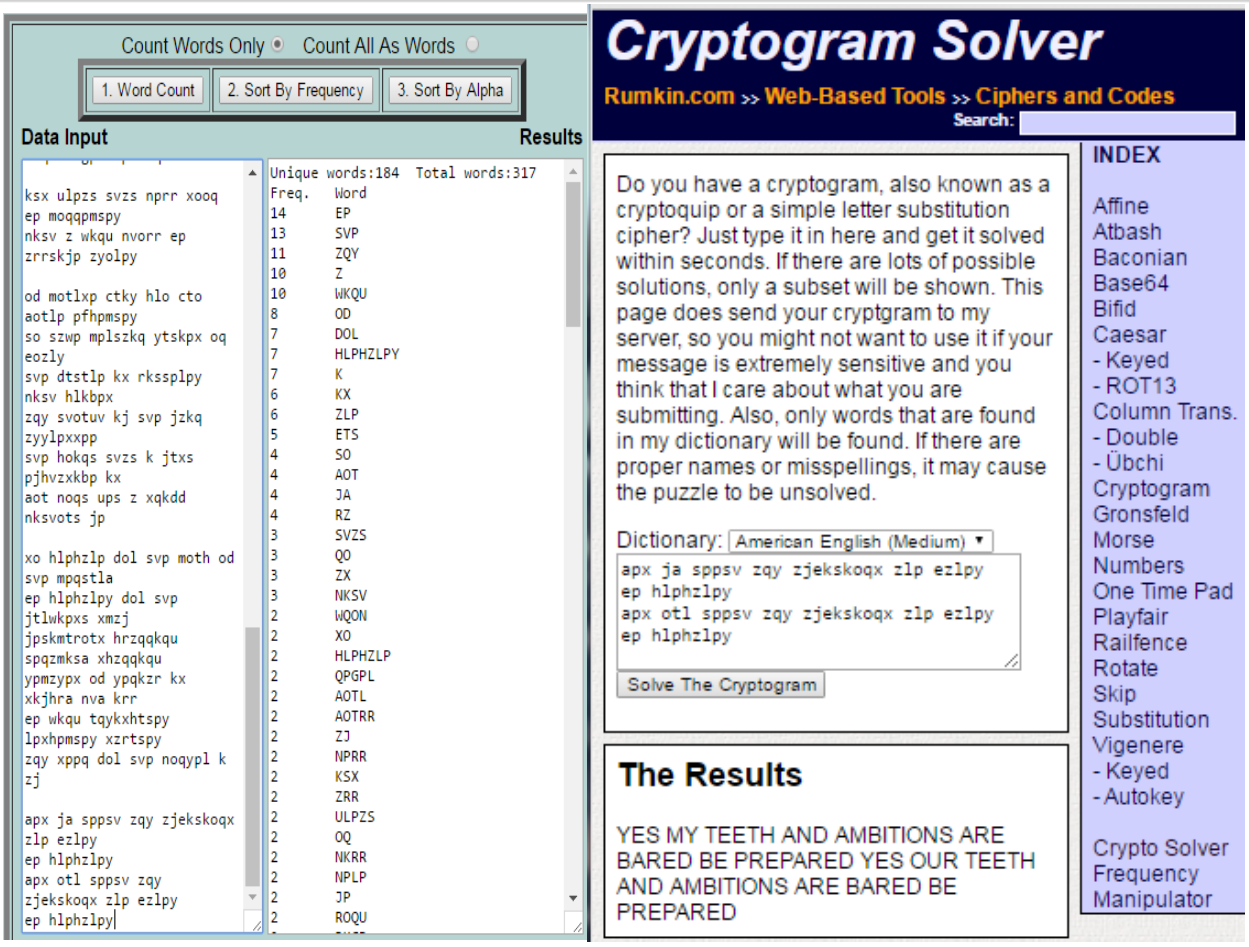
```
of to in it is be as at so we he by or on do
```

The most frequent doubly repeated letters are:

```
ss ee tt ff ll mm oo
```

The most frequent 3-letter words are:

```
the and for are but not you all
```



1.3.5 Tricky& Suggestion

1.3.5.1 Tricky

i. For the Caesar Cipher(ROT-X), though they have 25 kinds of methods for the text (X = 1~25), there will be only one X make sense.
ii. The letter frequency might be change according different text. The longer the text, the letter frequency will be more similar with the real distribute.
iii. Pay attention to the different combination of letter presentation.
iv. Sometimes the longer the given file is, the easier to be solved, since the long file will tend to the same situation with the real letter frequency.

1.3.5.2 Problem Design Suggestion

i. Use letter frequency and Caesar Cipher together.
ii. Combine several different letter presentations together.
iii. Give them a long encoded text file or just several short sentences.

1.4 Other Ciphers

1.4.1 Example 1: STPIR Cipher

Question description: http://ctfwriteups.blogspot.com/2016/02/internetwache-ctf-2016-crypto-pirat.html#more

1.4.2 Example 2: XOR Cipher

1.4.2.1 Description

In cryptography, the simple XOR cipher is a type of additive cipher, an encryption algorithm that operates according to the principles:

$A \oplus 0 = A$ $\qquad$ $A \oplus A = 0$

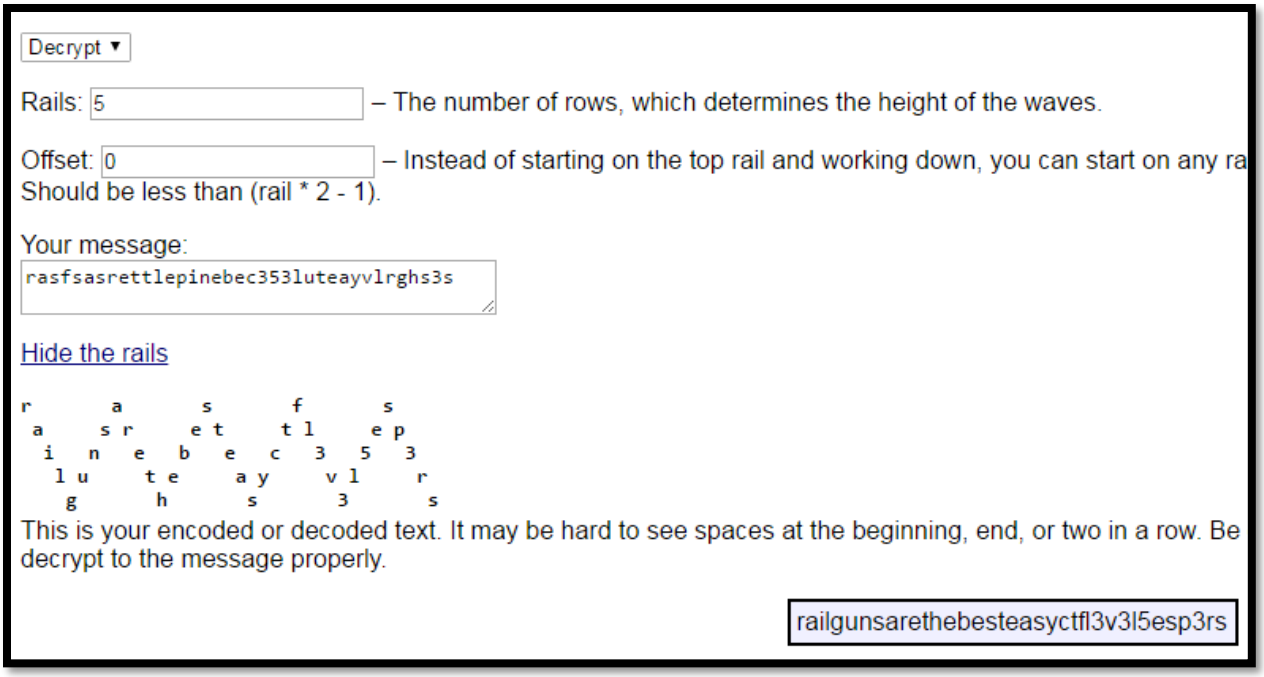
$(A \oplus B) \oplus C = A \oplus (B \oplus C)$ $\qquad$ $(B \oplus A) \oplus A = B \oplus 0 = B$

where denotes the exclusive disjunction (XOR) operation. This operation is sometimes called modulus 2 addition (or subtraction, which is identical), With this logic, a string of text can be encrypted by applying the bitwise XOR operator to every character using a given key. To decrypt the output, merely reapplying the XOR function with the key will remove the cipher.

1.4.3 Example 3: Rail Fence Cipher

Question description : http://writeups.easyctf.com/misaka_mikoto_50.html .

tool link: http://rumkin.com/tools/cipher/railfence.php



1.4.4 Tricky& Suggestion

1.4.4.1 Unsolved Questions

Question description link: (Unsolved)
https://picoctf.com/api/autogen/serve/encrypted.txt?static=false&pid=82a42d1d859d5c2140e8942848e5db0e (try to find the XOR Cipher to decode this file)

Try to find the length of XOR Cipher, use Kasiski examination get the substitution cipher first.

Solution link: https://ehsandev.com/pico2014/cryptography/repeated_xor.html

1.4.4.2 Tricky

i. Some Ciphers are rare.
ii. The hints are indirect.
iii. The length of XOR Cipher is not fixed; it could change according different file.

1.4.4.3 Problem Suggestion

i. Tyr to get some new cipher
ii. Try to make several different cipher together
iii. Try to make the hints indirect.

1.5 Useful Tool/Website Recommendation

Extract PDF file online: http://www.extractpdf.com/

Auto Convert online: https://conv.darkbyte.ru/

Letter frequency reference: *https://picoctf.com/crypto_mats/index.html*

English letter frequency calculator: https://www.mtholyoke.edu/courses/quenell/s2003/ma139/js/count.html

I33T alphabet table: https://qntm.org/l33t

Auto Cryptogram solver: http://rumkin.com/tools/cipher/cryptogram-solver.php

Tool: Caesar Cipher ROT-X: http://planetcalc.com/1434/

Rail fence Quick decode: http://rumkin.com/tools/cipher/railfence.php

Kasiski examination: https://en.wikipedia.org/wiki/Kasiski_examination

Cipher Tools: http://rumkin.com/tools/cipher/

2 Reverse engineering

2.1 Overflow

2.1.1 Example 1: Integer overflow

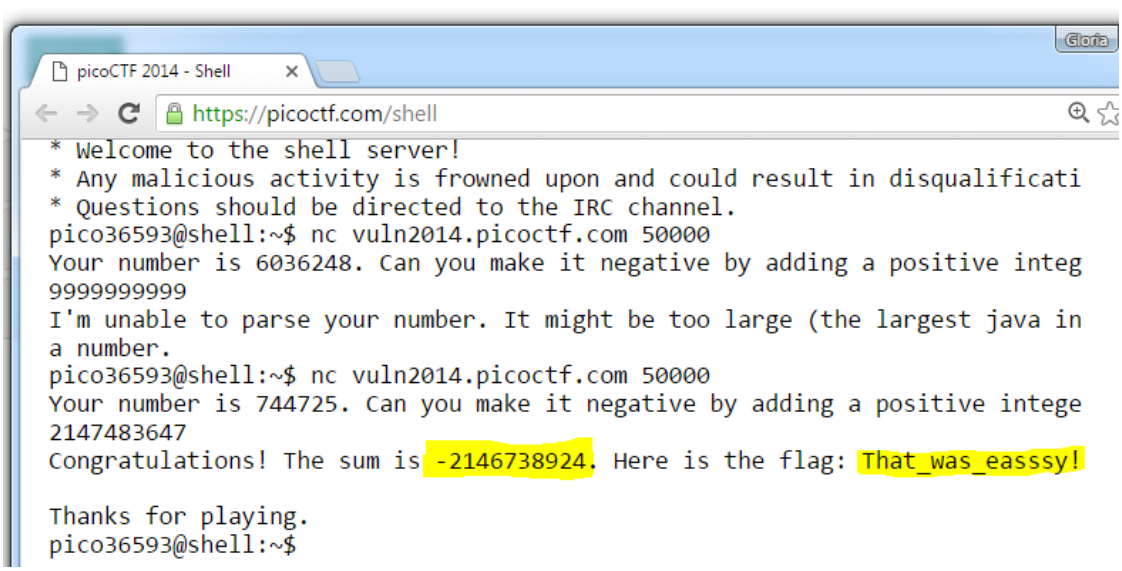
In computer programming, an integer overflow occurs when an arithmetic operation attempts to create a numeric value that is too large to be represented within the available storage space. For instance, taking the arithmetic mean of two numbers by adding them and dividing by two, as done in many search algorithms, causes error if the sum is too large to be represented, and hence overflows.

Question description below:

Is the sum of two positive integers always positive?

`nc vuln2014.picoctf.com 50000`

'nc' is the Linux netcat command. Try running it in the shell.



2.1.2 Example 4: Format string bugs

Format string vulnerabilities are a pretty silly class of bug that take advantage of an easily avoidable programmer error. If the programmer passes an attacker-controlled buffer as the argument to a printf, the attacker can perform writes to arbitrary memory addresses.

Question description and solution reference:
https://ehsandev.com/pico2014/binary_exploitation/format.html

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>

int secret = 0;

void give_shell(){
    gid_t gid = getegid();
    setresgid(gid, gid, gid);
    system("/bin/sh -i");
}

int main(int argc, char **argv){
    int *ptr = &secret;
    printf(argv[1]);

    if (secret == 1337){
        give_shell();
    }
    return 0;
}
```

2.1.2.1 More Examples about Format string bugs

Question 1: http://mslc.ctf.su/wp/plaidctf-2012-format-99-pwnables/

Question 2: http://mslc.ctf.su/wp/gits-2013-teaser-ppc/

Question 3: http://codearcana.com/posts/2013/05/02/introduction-to-format-string-exploits.html

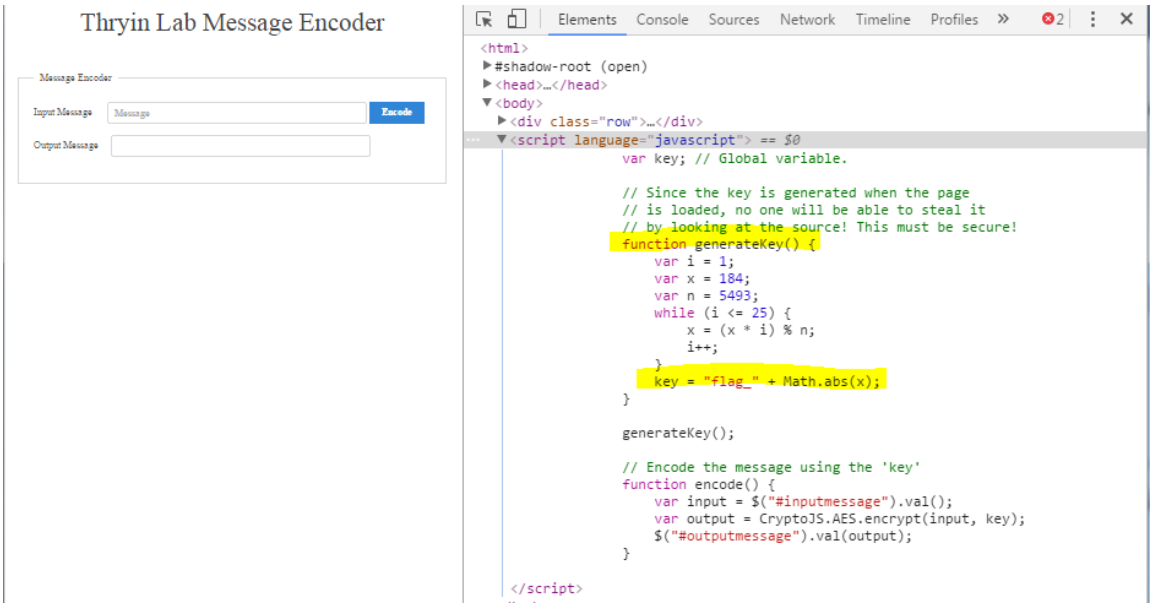
2.2 Binary Exploit 1: Pwnables

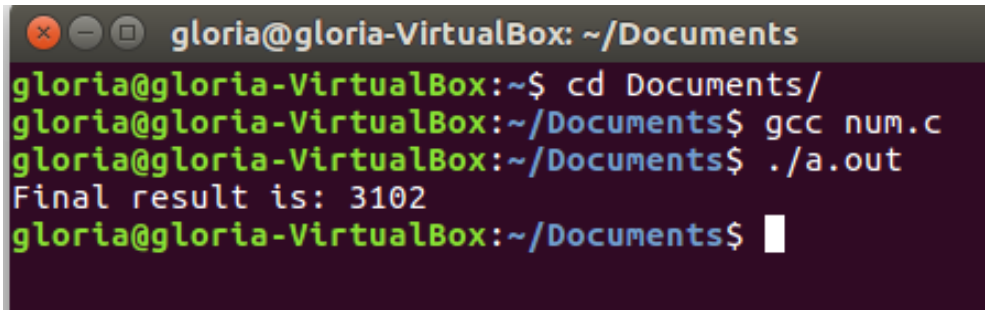
2.2.1 Description:

The program you received is running as a service on a remote machine. Reverse-engineer it, figure out its vulnerability, and use that to take exploit the remote service into serving you a flag.

2.2.2 Example 1: PHP

Tyrin Robotics Lab uses a special web site to encode their secret messages. Can you determine the value of the secret key?





According to the PHP file and the final result of JavaScript the Flag is : Flag_3102

2.2.3 Example 2: Code Reverse

Please login to picoCTF to use the webshell.

Binary Demo 1

`binary_demo1` is a program that resembles several of the picoCTF binary challenges.

When supplied with a correct command line argument, `binary_demo1` will invoke a shell. For real picoCTF challenges, the challenge binary will often be a **setgid binary**, which causes extra privileges to come with any shell invoked from within that binary (i.e. the privileges to read a `./flag` file). However, this demo binary is not setgid, so anything you could do from shell that gets invoked inside of it was already in fact achievable from shell you invoked `binary_demo1` from.

`binary_demo1` exercises some basic C syntax. Teaching C is beyond the scope of these educational materials, but most picoCTF challenges require nothing more than you might already know from AP CS Java or could pick up in a few hours with online C tutorials.

Reading the `binary_demo1.c`, it's not too hard to see that the "solution" to this demo is entering something like `./binary_demo1 this_is_the_password` on the command line. However, the actual argument you need to pass has non-ASCII characters on it.

I myself would use the python command `python -c 'print([hex((~ord(c))%256) for c in "the"])'` to check out exactly what those bitwise-complemented letters from `binary_demo1.c` actually are. Then, there are several ways to pass command-line arguments involving non-ASCII characters:

- *Bash escapes*: `./binary_demo1 $'this_is_\x8b\x97\x9a_pass\x99word'`

  In the bash shell, passing a string like `$'string'` allows backslash escape sequences to be used in the string, such as the hex codes shown above.

- *Command substitution*: `./binary_demo1 "$(cat /tmp/myfile)"`

  In any standard shell, you can use the output of another command as a command line argument like so: `"$(command arg1 arg2 ...)"`. In the above case, we're using the contents of file `/tmp/myfile` as our argument via the `cat` command. (You can type `man cat` on the terminal to get more information about the `cat` command).

- *Execing via python, perl, ruby, etc.*

  If you're comfortable in a scripting language, most have some way of invoking a binary with specific args. For example, in python you could

```
>>> from os import execl
>>> execl('./binary_test1','./binary_test1','this_is_\x8b\x97\x9a_pass\x99word')
```

Heads up: For many "exec" like functions (including in languages other than python), you'd seemingly need to pass `./binary_test1` twice: once to state where the `binary_test1` binary is, and the second time to set the `argv[0]` value that `binary_test1` will see. (Programs don't usually care what argv[0] actually is, but something still needs to be there.)

For additional information about C or the bash shell, the internet is a great resource ^_^

Source Code: binary_demo1.c

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[]) {
  printf("Hi, I'm %s\n", argv[0]); // The first "argument" is the path-to-self

  if(argc != 2) { // argc is how long argv is, including that path-to-self
    puts("You need to give me exactly one argument!");
    return -1;
  }

  // if statements, character and string constants, && and ||, and
  // comparisons work a lot like Java or any other language.
  if(argv[1][0] == 't') {
    if(argv[1][1] == 'h' && argv[1][2] == 'i' && argv[1][3] == 's') {

      // In C, characters are just numbers from 0 to 255.
      if(argv[1][4] == 95 && argv[1][5] == 105 && argv[1][6] == 0x73) {
        puts("half way there!");
        if(argv[1][7] != 0x5f) return 0;

        // Since characters are just numbers, we can bit-flip them!
        if(argv[1][8] != ~'t' || argv[1][9] != ~'h' || argv[1][10] != ~'e') {
          puts("noooo!");
          return 0;
        }

        // Lastly, "strings" are really just pointers to characters.
        // argv[1][11] is the 12th character of argv[1]. So, &argv[1][11]
        // points at that 12th character, which in C is basically just
        // the argv[1] string from the 12th character onwards.
        if(strcmp("_pass\x99word", &argv[1][11]) == 0) {
          // strcmp returns 0 if the strings match, and non-zero if
          // they don't. You can type `man strcmp` (without backticks)
          // at the terminal to get more detailed information on strcmp.
          puts("yep, that's it!");
          system("sh -i");
          return 0;
        }
      }
    }
  }
  puts("nope!");
  return 0;
}
```

2.2.4 Tricky& Suggestion

2.2.4.1 Unsolved Questions

Question 1: https://cedricvb.be/post/reverse-engineering-the-hitb-binary-100-ctf-challenge/#prettyPhoto

Question 2: http://challenges.hackucf.org/practice/meet2/#collapseFour

Training Questions Link: https://blog.skullsecurity.org/category/reverse_engineering

2.3 Binary Exploit 2: Crackmes

2.3.1 Description:

These programs include anti-reversing measures, like anti-debugger instructions, code obfuscation, or even dual-use opcodes. There's a few ways the flag could be embedded - the program could implement a particular password check, and the string that is accepted is the flag, or perhaps the program calculates the decryption of an encrypted flag, but requires rewriting certain instructions so that it performs the decryption.

2.3.2 Example:

The police need help decrypting one of your father's files. Fortunately, you know where he wrote down all his backup decryption keys as a backup (probably not the best security practice). You are looking for the key corresponding to `daedaluscorp.txt.enc`. The file is stored on the shell server at `/problems/grepfriend/keys`.

```
pico36593@shell:~$ grep "daedaluscorp" /problems/grepfriend/keys
daedaluscorp.txt.enc        b2bee8664b754d0c85c4c0303134bca6
pico36593@shell:~$
```

2.3.3 Tricky& Suggestion

2.3.3.1 Tricky

i. Need a long time to analysis the program
ii. Quite familiar with the GDB command

2.4 Useful Tool/Website Recommendation

Webpage source page check: Ctrl + U

Webpage resource Check: Ctrl + Shift + i

Environment: Linux/Unix

Assemble language: https://en.wikipedia.org/wiki/Assembly_language

AT& T vs. Intel syntax: https://en.wikipedia.org/wiki/X86_assembly_language#Syntax

Machine code: https://en.wikipedia.org/wiki/Machine_code

GNU Assembler: https://en.wikipedia.org/wiki/GNU_Assembler

Format String Bug: https://www.owasp.org/index.php/Format_string_attack

Introduction of format string exploits:

http://codearcana.com/posts/2013/05/02/introduction-to-format-string-exploits.html

3 Web exploitation (most popular are PHP & SQL)

3.1 Core problem

The majority of attacks against web application involved sending input to the server which is crafted to cause some event that was not expected or desired by the designer.

A. The users can submit any input, include:
   i. Request parameters, cookies, HTTP headers
   ii. Send request in any sequence
   iii. Not restricted to using only one web browser
B. The defense mechanisms comprise the following core elements:
   i. Handing user access: Authentication, Session Management, Access Control

ii. Handing user input: Varieties of Input, Approaches to Input Handing, Boundary Validation
iii. Handing user attackers: Handing Errors, maintaining audit logs, alerting administrators, reacting to attacks.
iv. Managing the application

3.2 SQL Injection

3.2.1 Description: Based on "="/ “- -”

Submit usernames and passwords that modify the SQL query – This is SQL injection! It is because SQL don't 'escape' the input. If we made a ‘and - - in the input, it will end the SQL early and ignore the rest queries. The ' closes the string for username, and -- makes the SQL query end, so it doesn't check for password

3.2.2 Example 1: Log injection

This is an example of mid-level SQL injection, Hints: you should get the password of a user with admin privileges. (Below is the login.php source code)

```php
<?php
include "config.php";
$con = mysqli_connect("localhost", "sql1", "sql1", "sql1");
$username = $_POST["username"];
$password = $_POST["password"];
$debug = $_POST["debug"];
$query = "SELECT * FROM users WHERE username='$username' AND password='$password'";
$result = mysqli_query($con, $query);

if (intval($debug)) {
  echo "<pre>";
  echo "username: ", htmlspecialchars($username), "\n";
  echo "password: ", htmlspecialchars($password), "\n";
  echo "SQL query: ", htmlspecialchars($query), "\n";
  if (mysqli_errno($con) !== 0) {
    echo "SQL error: ", htmlspecialchars(mysqli_error($con)), "\n";
  }
  echo "</pre>";
}

if (mysqli_num_rows($result) !== 1) {
  echo "<h1>Login failed.</h1>";
} else {
  echo "<h1>Logged in!</h1>";
  echo "<p>Your flag is: $FLAG</p>";
}

?>
```

Then we can just use: admin’ - -, then we can login successfully.

3.2.3 Example 2: login injection

http://web2014.picoctf.com/injection1/

first we try the basic injection query, use admin' or 1 = 1 --, it shows:

```
username: admin'  or 1 = 1 --
password: admin'  or 1 = 1 --
SQL query: SELECT * FROM users WHERE username='admin'  or 1 = 1 --' AND
password='admin'  or 1 = 1 --'
SQL error: You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near
'admin'  or 1 = 1 --'' at line 1
```

Login failed.

So we can analysis how the query work. It shows we have an error, and show the - - actually doesn't work, and an extra ' makes this error. Then we modify our query as: admin' or '1' =' 1 and make password as null (or you can try it as any value it seems)

then it works!

```
username: admin'  or '1' =' 1
password:
SQL query: SELECT * FROM users WHERE username='admin'  or '1' =' 1' AND password=''
```

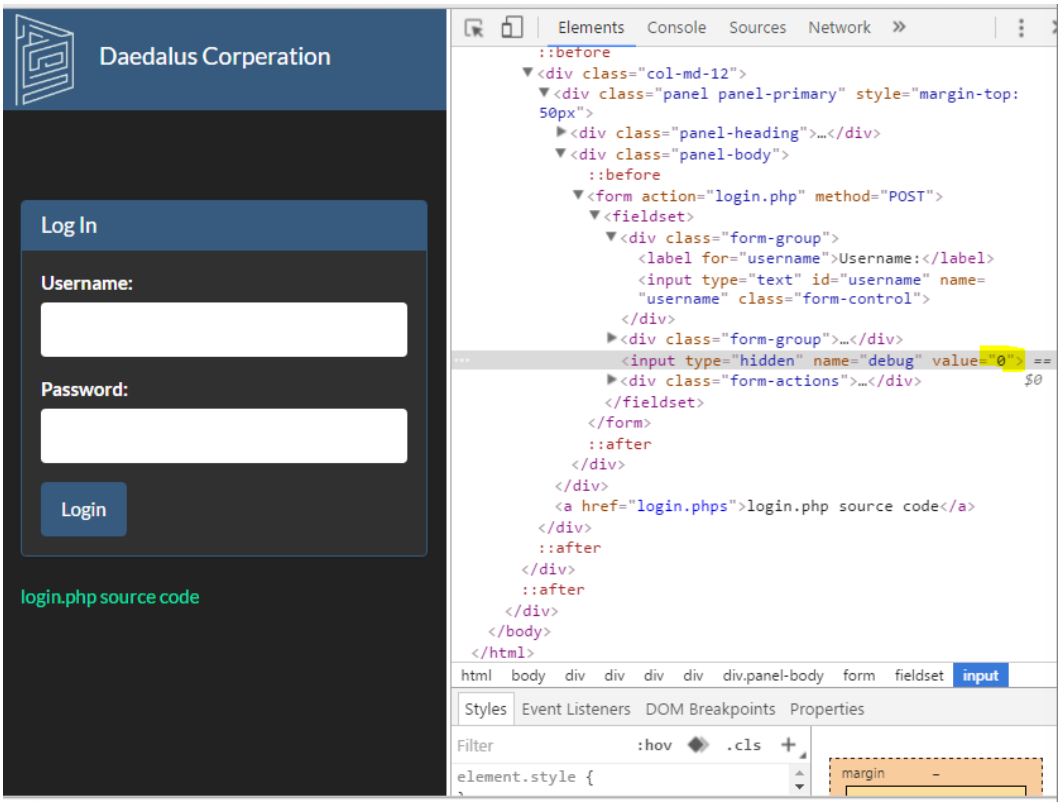
Logged in!

Your flag is: flag_vFtTcLf7w2st5FM74b

3.2.4 Example 2: Command Injection

http://web2014.picoctf.com/injection2/

First check the source code – login.php file



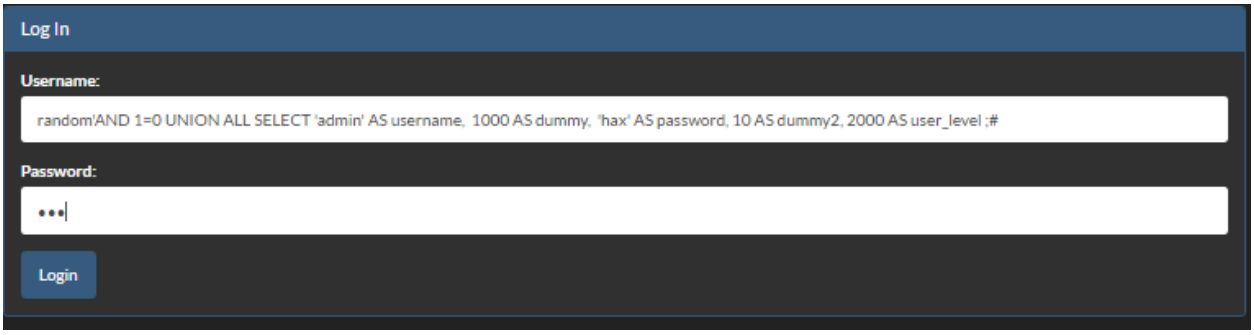
Change this value to 1 to make all the SQL query show on screen.

Then we try the common SQL injection query, “admin' or '1' =' 1 ” But it shows:

```
username: admin'  or '1' =' 1
password:
SQL query: SELECT * FROM users WHERE username='admin'  or '1' =' 1'
```

Login failed.

So we know how the SQL works with the input username. Then we append UNION ALL and then our fake select statement. We put in things we think will get past our checks, so : UNION ALL SELECT 'admin' AS username, 'hax' AS password, ‘1337’ AS user_level ;# will work, and make sure use the same ‘hax’ as your password for the website. Then we get the flag.



You can use different password, but just keep the password in the SQL query in the username is the same with the input password.

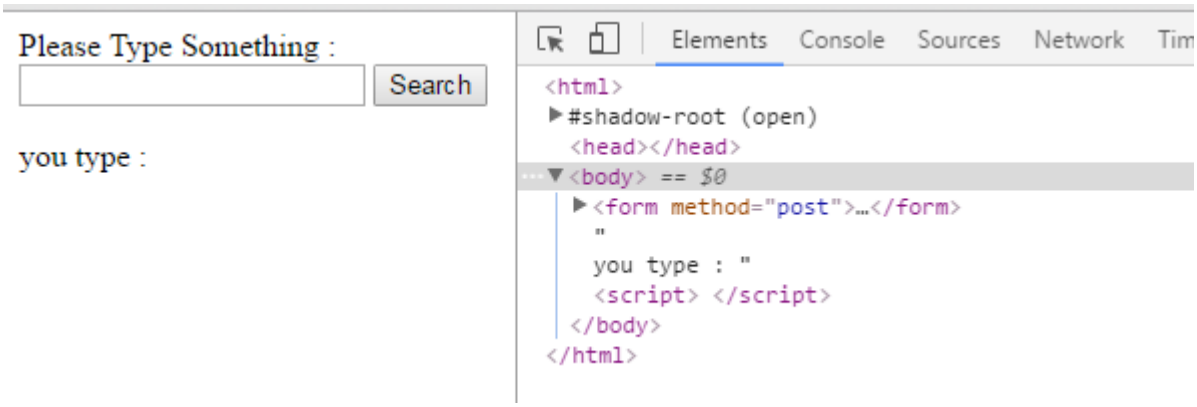
3.3 XSS (Cross-Site Scripting)

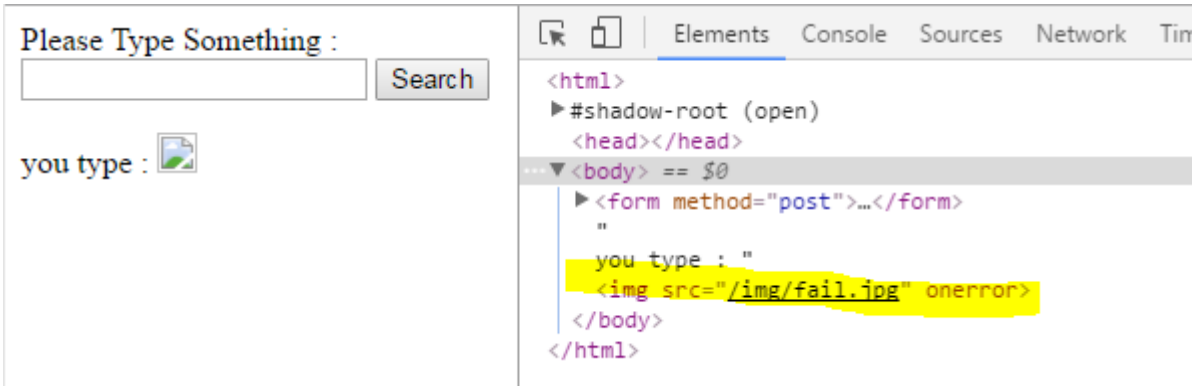
3.3.1 Description

Cross-site scripting (XSS) is a type of computer security vulnerability typically found in web applications. XSS enables attackers to inject client-side scripts into web pages viewed by other users.

3.3.2 Example 1: Change source code

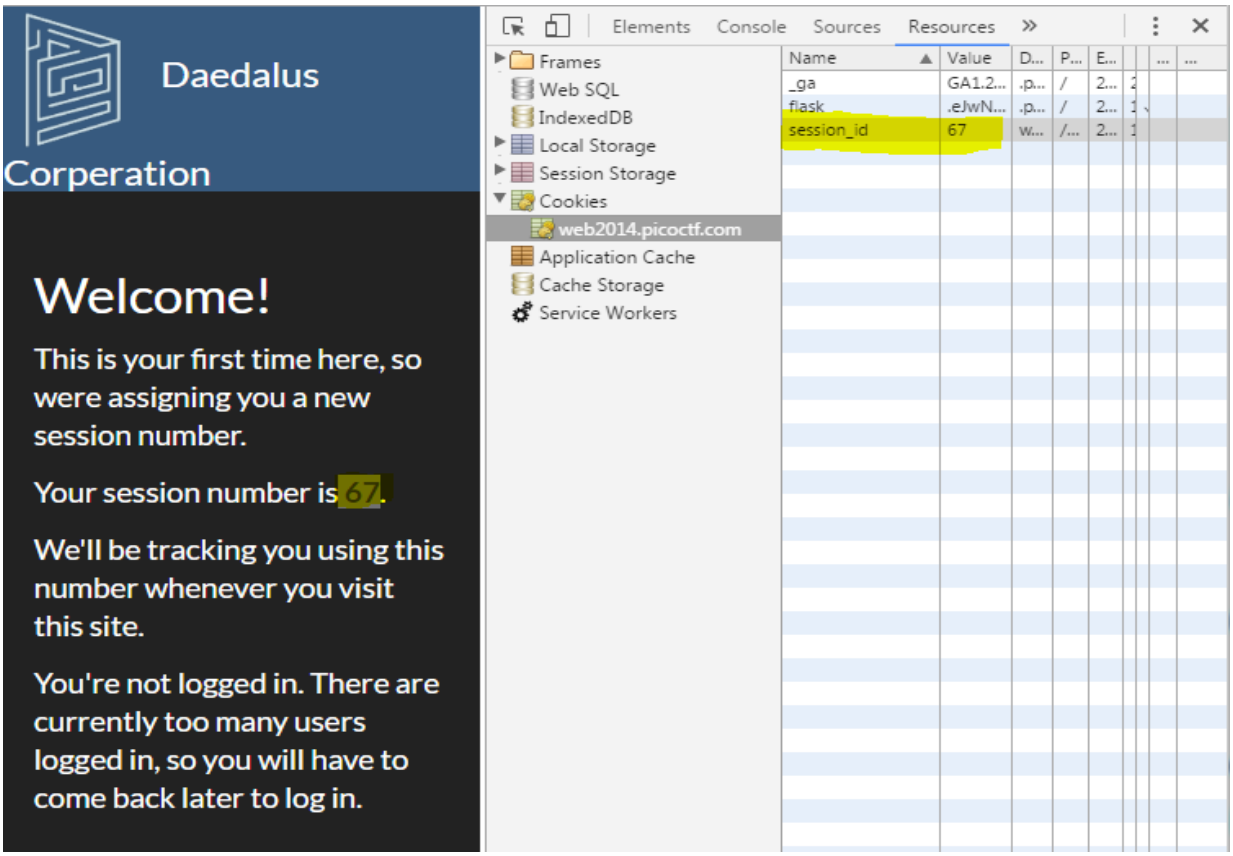
The first screenshot is for the original code, it shows “you type” then <script>…</script>, then after I type : <img src="/img/fail.jpg" onerror="javascript:alert('YoU aRe hAcKeD')"> in the input area, the source code changes to the second screen shot, it shows the input is considered as a html code in the source code, and it tried to get the picture /img/fail.jpg.

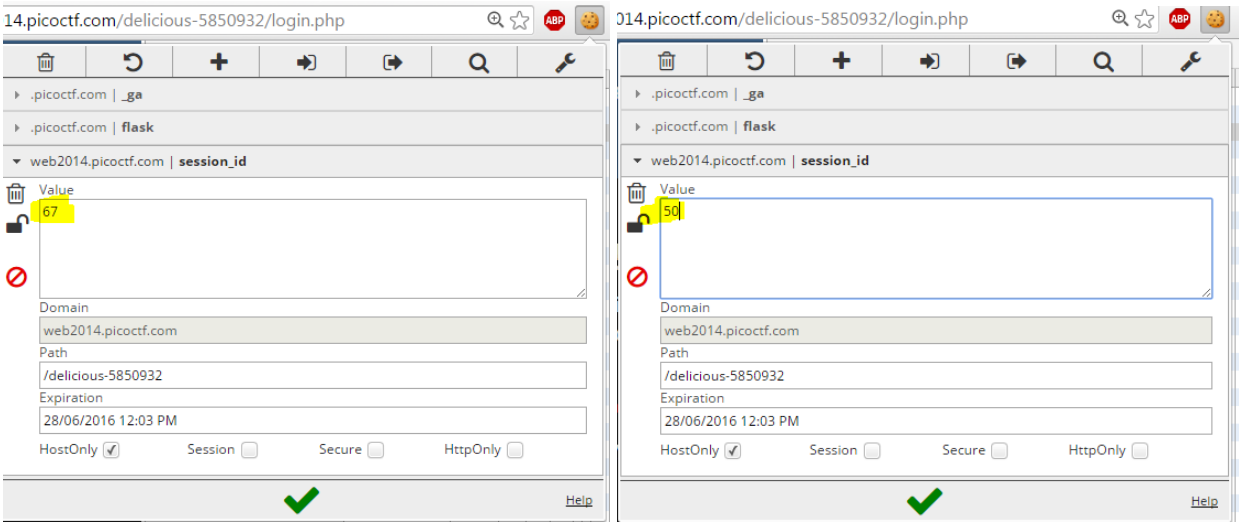




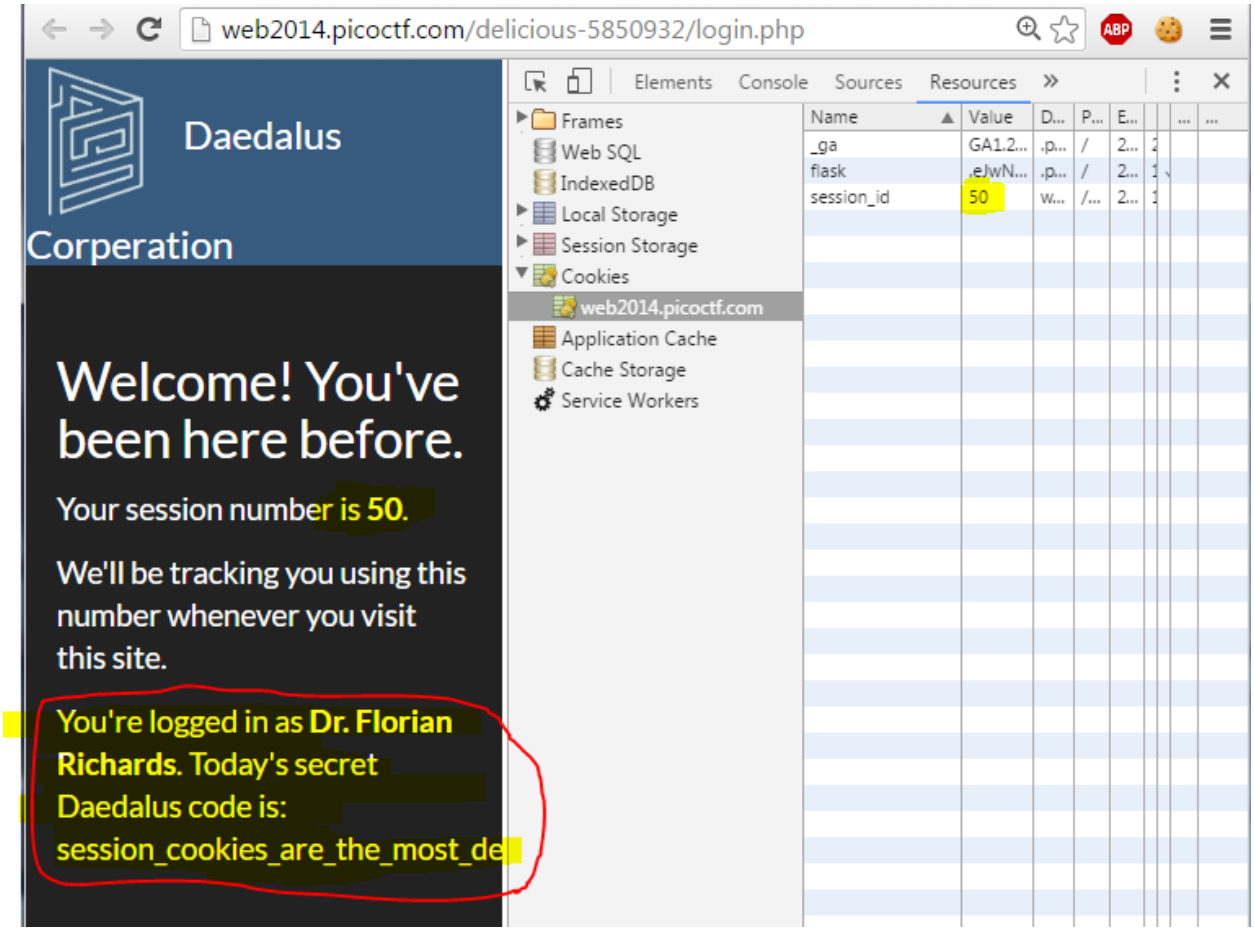
3.3.3 Example 2: Change Cookies

Problem description: http://web2014.picoctf.com/delicious-5850932/login.php





Change the section_id from 67 to a smaller number such as 50(try some other number but smaller than 67). And then refresh the webside, it will send your changes cookie to the server, then you will be concerdered as the person whoes section id is 50. And in the new page, we get the information about this person's name and maybe some other information if it happened in the real world.



3.4 Useful Tool/Website Recommendation

For Chrome, EditThisCookie

For Firefox, Cookie Manager

SQL MAP & SQL Injection cheat sheet:

http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet

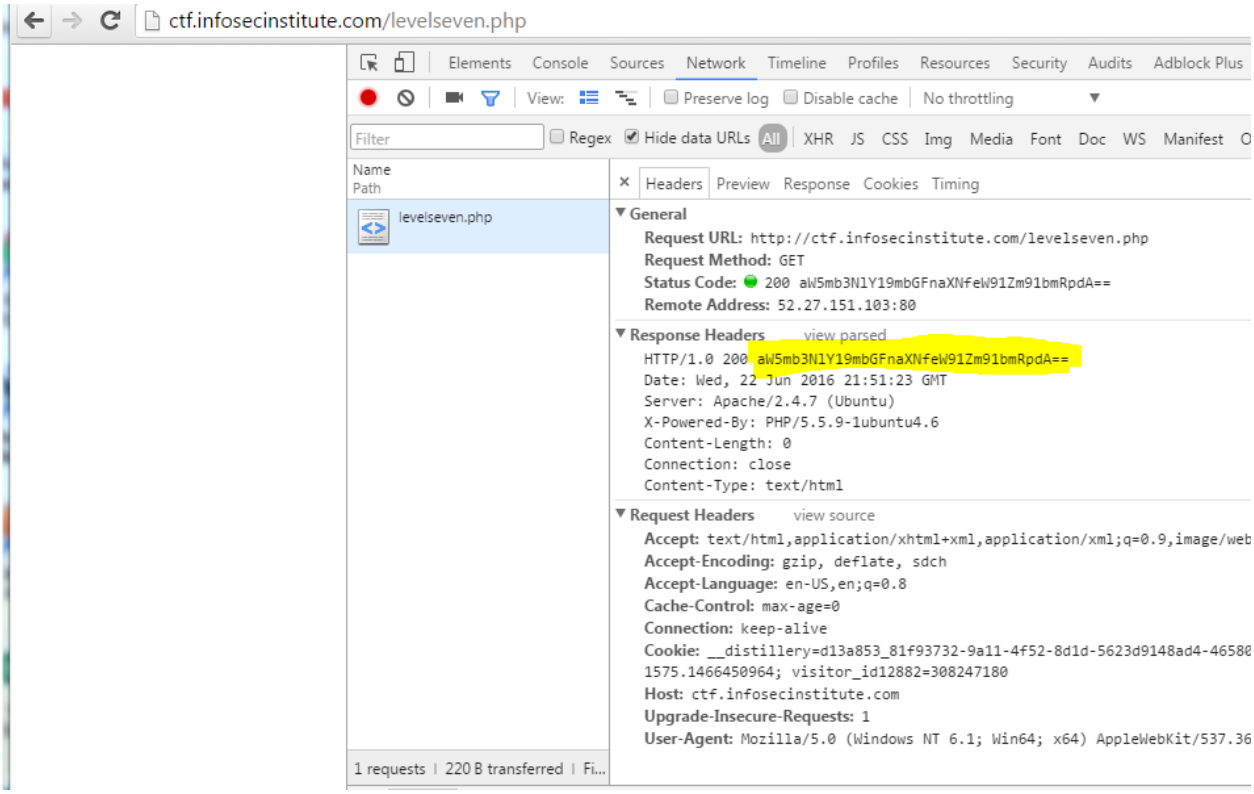
4 Steganography

4.1 Hidden flag (PHP)

4.1.1 Description: Hide a secret message in some plain sight

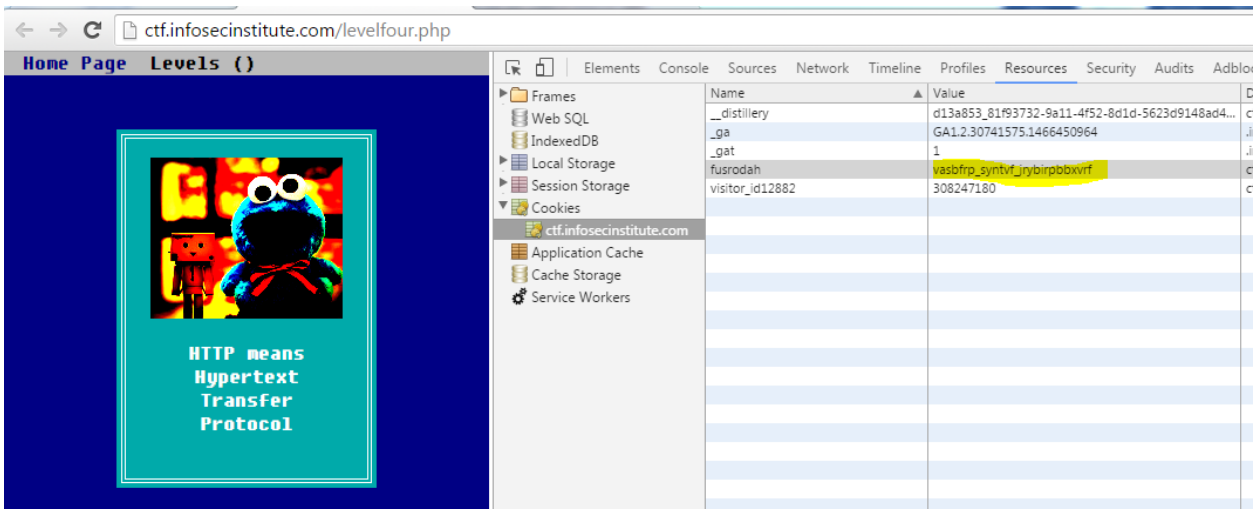
4.1.2 Example 1: Header

http://ctf.infosecinstitute.com/levelseven.php



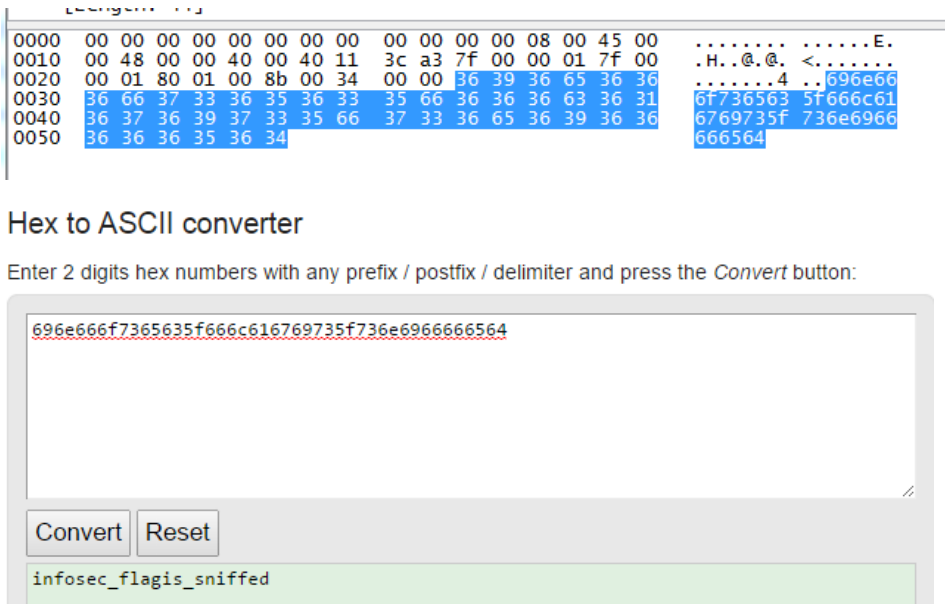
4.1.3 Example 2: Cookie

http://ctf.infosecinstitute.com/levelfour.php



4.1.4 Example 4: Download file

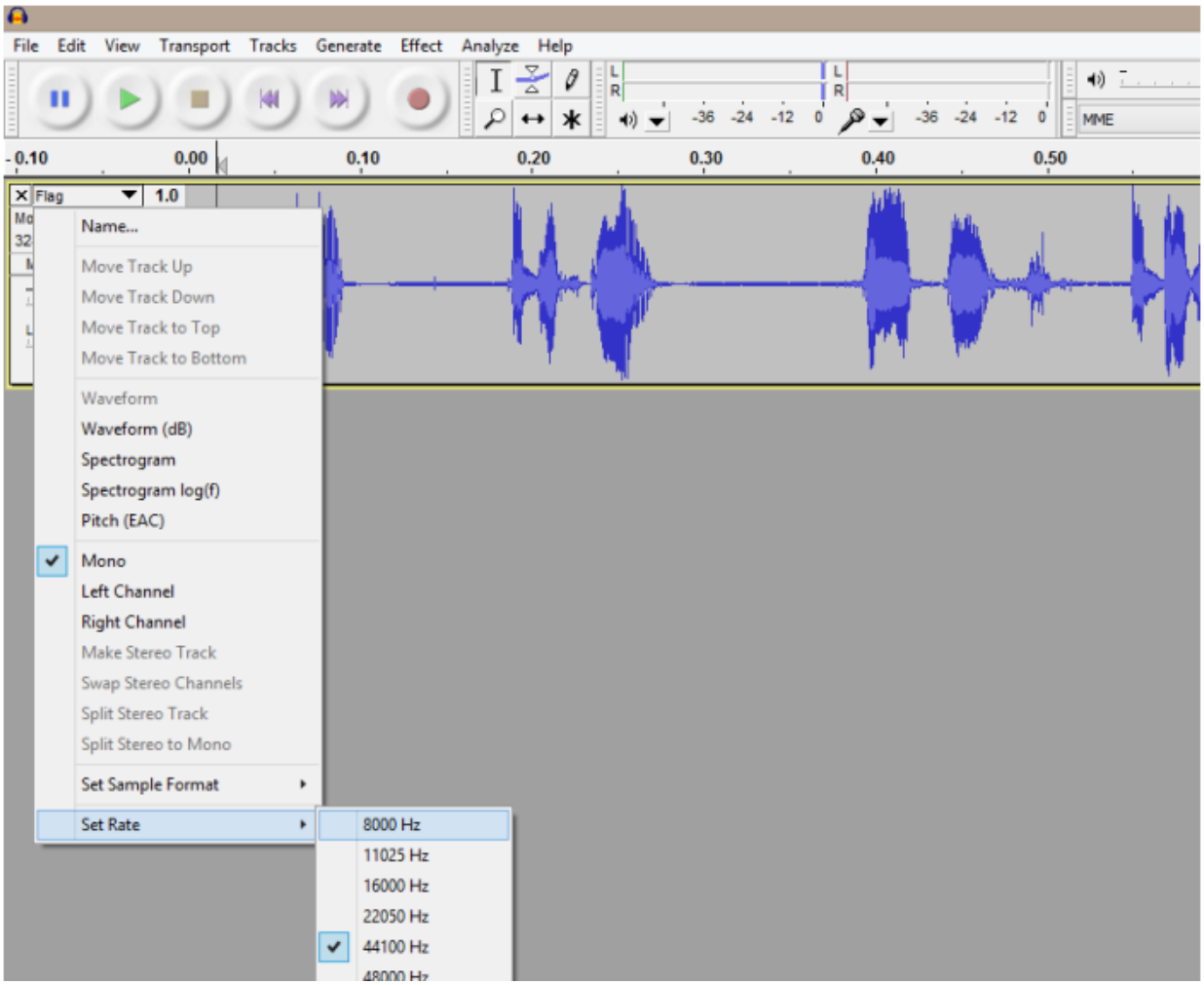
https://travisroyer.wordpress.com/2015/03/14/n00bs-ctf-labs-level-6-solution/



Or for here, we can design the convert part as complex as we can.

4.1.5 Example 5: Sound file

Question link: http://ctf.infosecinstitute.com/levelten.php



The original rate higher than the normal rate, and hide flag as sounds inside. Then the reduce the rate to get the sound flag. After reduce the rate, we can hear the voice: infosec_flagis_sound

4.1.6 Tricky& Suggestion

here are so many place they/we could hide the flag. So we have to so a lot of work relate with search. We have to try different kinds of methods about any potential place. And also we can hide our flag deep in the code, make the attackers have to spend a lot of time to solve it.

4.2 Forensics

4.2.1 Description

The file can be different type file with its expand name, for example, a PDF file can be .exe file, and a .zip file also can be a .jpeg file. Don't believe files expand name

4.2.2 Example: Fake files

Use Linux command: file/strings/ pngcheck and so on

We are given a dog PNG picture.

Running `pngcheck` on the picture reveals that it is not a PNG, however:

```
$ pngcheck a038218191c05846099054a9f21ff22a.png
a038218191c05846099054a9f21ff22a.png  this is neither a PNG or JNG image nor a MNG stream
ERROR: a038218191c05846099054a9f21ff22a.png
```

It's actually a JPEG:

```
file a038218191c05846099054a9f21ff22a.png
a038218191c05846099054a9f21ff22a.png: JPEG image data, JFIF standard 1.01
```

4.3 Useful Tool/Website Recommendation

File Signature List: http://www.garykessler.net/library/file_sigs.html

Audio editor: Audacity http://www.audacityteam.org/download/

5 Miscellaneous

5.1 Quiz questions

reference: …\fbctf\demo_levels\quiz_examples.md

5.1.1 Description:

These are often lower-point value challenges, include some basic network definition questions, some questions about internet development history. And algorithm question such as ACM style questions.

5.1.2 Example: Short questions list

1. What's google.co.uk's SSL certificate public key size?

- 2048

2. What mechanism allows web servers to instruct web browsers to only interact with them using secure HTTP connections?

- HSTS

3. What protocol is described in RFC 2324?

- Hyper Text Coffee Pot Control Protocol

4. What novel served as inspiration for the HTTP "Unavailable for Legal Reasons" error code?

- Fahrenheit 451

5. What technique is used to prevent return-to-libc attacks? (4 letters)

- ASLR

5.2 Useful Tool/Website Recommendation

RSA Algorithm : https://simple.wikipedia.org/wiki/RSA_(algorithm)

Short quiz reference: fbctf foler path …\fbctf\demo_levels\quiz_examples.md