

Level1

- **Precondition:**

Initial flag given by twitter's DM: ONA{We1come_To_0napsis_Ch4llenges}

- **Description:**

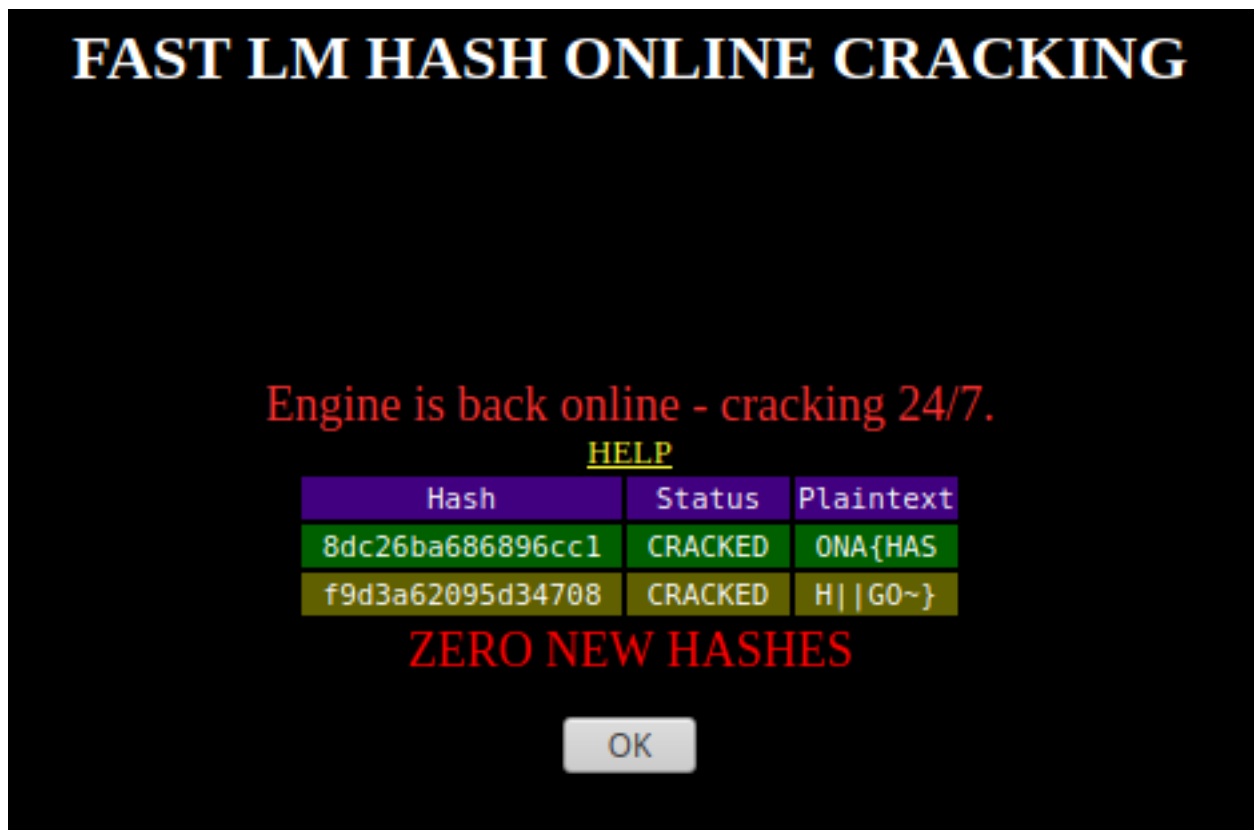
Before you can start playing our games, we need you to decrypt the secret flag from game 1. One of our developers hid it using this python script, but now he can't recover it. It seems like the password will change in each run...

- **Analysis:**

After checking the code I realized the hash at the beggning of the script is an LMHASH

- **Solution:**

I proceded to decrypt it through an online service: <http://rainbowtables.it64.com/>



- **Gotten flag:**

ONA{HASH||GO~}

Level2

- **Precondition:**

```
previous flag: ONA{HASH||GO~}
```

- **Description:**

I love to play music games, but my friends hate when I play on their computers. They say I knock too loud... but i have to be precise, cause I'm knocking on heaven's doors!!!

- **Analysis:**

The folder contains a .py script, wich is byte-compile, and a hidden plain text file that is a base64 string

```
androlde@d3br4:~/ctf/onapsis/filesBkp/Lvl2$ ls -la
total 12364
drwxr-xr-x 2 androlde androlde   4096 Aug 13 04:47 .
drwxrwxr-x 8 androlde androlde   4096 Aug 28 22:31 ..
-rw-r--r-- 1 androlde androlde   3394 Aug  3 17:27 knocking.py
-rw-r--r-- 1 androlde androlde 12640428 Aug  3 17:07 .music
-rw-r--r-- 1 androlde androlde    39 Aug 13 04:40 Requeriments.txt
androlde@d3br4:~/ctf/onapsis/filesBkp/Lvl2$ file knocking.py .music
knocking.py: python 2.7 byte-compiled
.music:      ASCII text, with very long lines, with no line terminators
androlde@d3br4:~/ctf/onapsis/filesBkp/Lvl2$ head -c 100
^C
androlde@d3br4:~/ctf/onapsis/filesBkp/Lvl2$ head -c 100 .music
SUQzBAAAFYmZVRBTEIAAAXAAAB//5FAE4AVABFAFIAIABIAGUAcgBvAFQRORTEAAAAZAAB//5zAG8AbQBtACAASABHAGMAawBl
androlde@d3br4:~/ctf/onapsis/f
ilesBkp/Lvl2$
androlde@d3br4:~/ctf/onapsis/filesBkp/Lvl2$ tail -c 100 .music
AAAAAAAAAAAAAU5URVIGSGVybwAAAAAAAAAAAAAAAAAAAAAAAAAAAmjAyMFNHVNnirzhnUjNWNWNSd2dhr1Z5WLNCaGntVWcAAP8=
androlde@d3br4:~/ctf/onapsis/f
ilesBkp/Lvl2$
```

- **Solution:**

After decoding the b64 encoded file I found it was a mp3 file with a b64 string in its metadata comment

[illegible]

The b64 string resulted being the instructions to play the game coded in python. I don't like to play so I continued with the decompilation of the script.

In a simple research I found a python module to decompile the byte-compiled script included in the 7z file.

```
$ pip2 install uncompyle6
```

The python version installed in my box wasn't included in the list of supported python interpreters so I added it hoping the magic numbers for the bytecodes haven't changed.

I added my python version here </home/andro1de/.local/lib/python2.7/site-packages/xdis/magics.py> and run the tool again. (Note: uncompyle6 needs pyc extensions so I changed the name)

```
$ uncompyle6 knocking.pyc
```

In the following code box you will find the script decompiled with the previous command

```
# uncompyle6 version 3.7.3
# Python bytecode 2.7 (62211)
# Decompiled from: Python 2.7.18rc1 (default, Apr 7 2020, 12:05:55)
# [GCC 9.3.0]
# Embedded file name: knocking.py
# Compiled at: 2020-08-03 17:26:57
import os, vlc
from datetime import datetime
import atexit, base64, time

dat = [
    [61, 'Z'], [61, '4'], [61, 'm'], [68, 'x'], [68, 'b'], [68, 'h'], [75, 'Z'], [75, 'a'], [76, 'y'], [82, 'B'],
    [82, 'd'], [83, 'P'], [148, 'T'], [148, '8'], [149, 'k'], [155, 'F'], [156, 'c'], [156, '7'], [162, 'S'],
    [163, '9'], [163, '2'], [170, '4'], [170, '2'], [170, 'w'], [234, 'K'], [235, '0'], [235, 'G'], [241, 't'],
    [242, '7'], [242, 'p'], [249, 'b'], [249, '5'], [250, 'l'], [256, '9'], [256, 'b'], [257, 'J'], [263, 'b'],
    [264, '4'], [264, 'n'], [271, 'R'], [271, 'd'], [271, 'S'], [278, 'M'], [278, 'f'], [278, 'F'], [285, '9'],
    [285, 'e'], [286, 'I'], [292, 'Z'], [292, '7'], [293, 'T'], [299, 'R'], [300, '0'], [300, '2'], [306, 'Z'],
    [307, 'a'], [307, 'U'], [314, '5'], [314, '2'], [314, '9']]

def exit_handler():
    f = open('.KnockinOnHeavensDoor.mp3', 'w')
    f.write('')
    f.close()

def decode(music, outfile):
    f = open(music, 'rb')
    bs64 = f.read()
    f.close()
    out = base64.b64decode(bs64)
    f = open(outfile, 'w')
    for i in out:
        f.write(i)

    f.close()

def knock(initial, tr):
    k = raw_input()
    b = datetime.now()
    delta = b - initial
    delta = abs(delta.seconds - tr[0])
    if delta < 2:
        return tr[1]
    return False

def chrKnock(initial, tn):
    s = ''
    a = knock(initial, dat[tn])
    if a == False:
        return False
```

```

b = knock(initial, dat[(tn + 1)])
if b == False:
    return False
c = knock(initial, dat[(tn + 2)])
if c == False:
    return False
if a != False:
    s = s + a
    if b != False:
        s = s + b
        if c != False:
            s = a + c
        return s
return False

print 'Hello rock fan!!! Here is a game you will like'
print 'Give me a minute to generate and load the chords...'
try:
    decode('.music', '.KnockinOnHeavensDoor.mp3')
except:
    print 'some files are missing, please be sure to decompress all the game files in the same folder and play the game!'
    exit()

atexit.register(exit_handler)
print 'Ok, im ready. Hit Intro to start playing the music and dont lose the pitch:'
raw_input()
player = vlc.MediaPlayer('.KnockinOnHeavensDoor.mp3')
player.play()
ini = datetime.now()
f = ''
for i in range(20):
    out = chrKnock(ini, i * 3)
    if out != False:
        f = f + out
    else:
        print '\nYou Failed! You are not listening to the music... Practice some more and try again!'
        print ''
        exit()

time.sleep(7)
print '\nCongratulations, you Rock!!! Here you have your reward:'
print str(base64.b64decode(f))
# okay decompiling knocking.pyc

```

After some analysis of the code, I removed all the code related to the loading of the music and I also modified the following function in order to get the flag without playing the game.

```

def knock(initial, tr):
    return tr[1]

```

Finally, executing the modified script, I got the flag

```

androlde@d3br4:~/ctf/onapsis/filesBkp/Lvl2$ python modified_knocking.py
Congratulations, you Rock!!! Here you have your reward:
flag ONA{Kn0(kin_IntR0_He4veN}
androlde@d3br4:~/ctf/onapsis/filesBkp/Lvl2$ 

```

- **Gotten flag:**

ONA{Kn0(kin_IntR0_He4veN)}

Level3

- **Precondition:**

previous flag: ONA{Kn0(kin_IntR0_He4veN)}

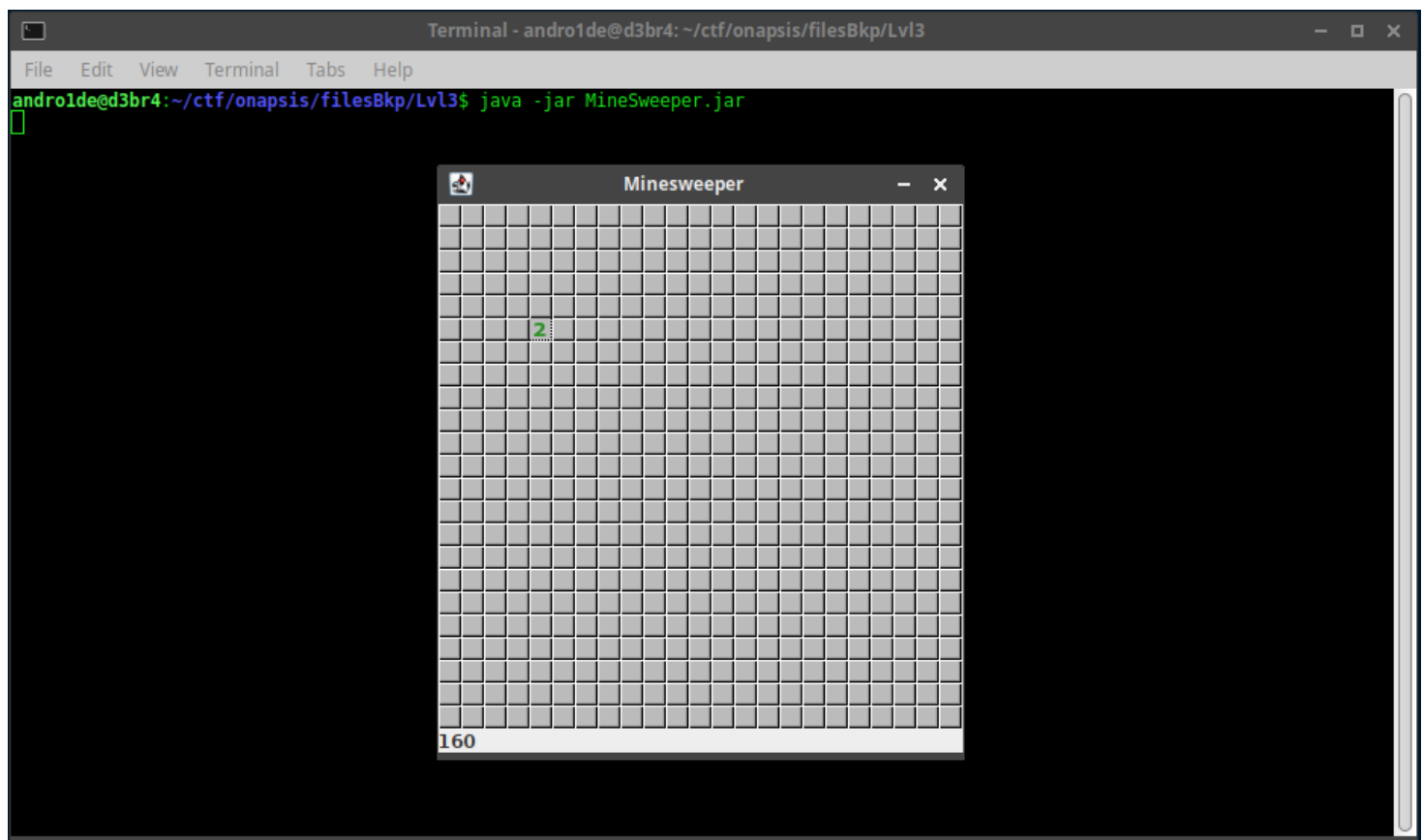
- **Description:**

A classic. Who did not play Minesweeper in their lives.

I just wish there weren't so many mines! And in such a small space. But the only way to get the flag is to solve it this way

- **Analysis:**

In the folder, I just found a jar file so I executed it.



Superficially, It seems that I just need to play and find all the mines. However, let's decompile the jar file, modify it and get a shorter way to win.

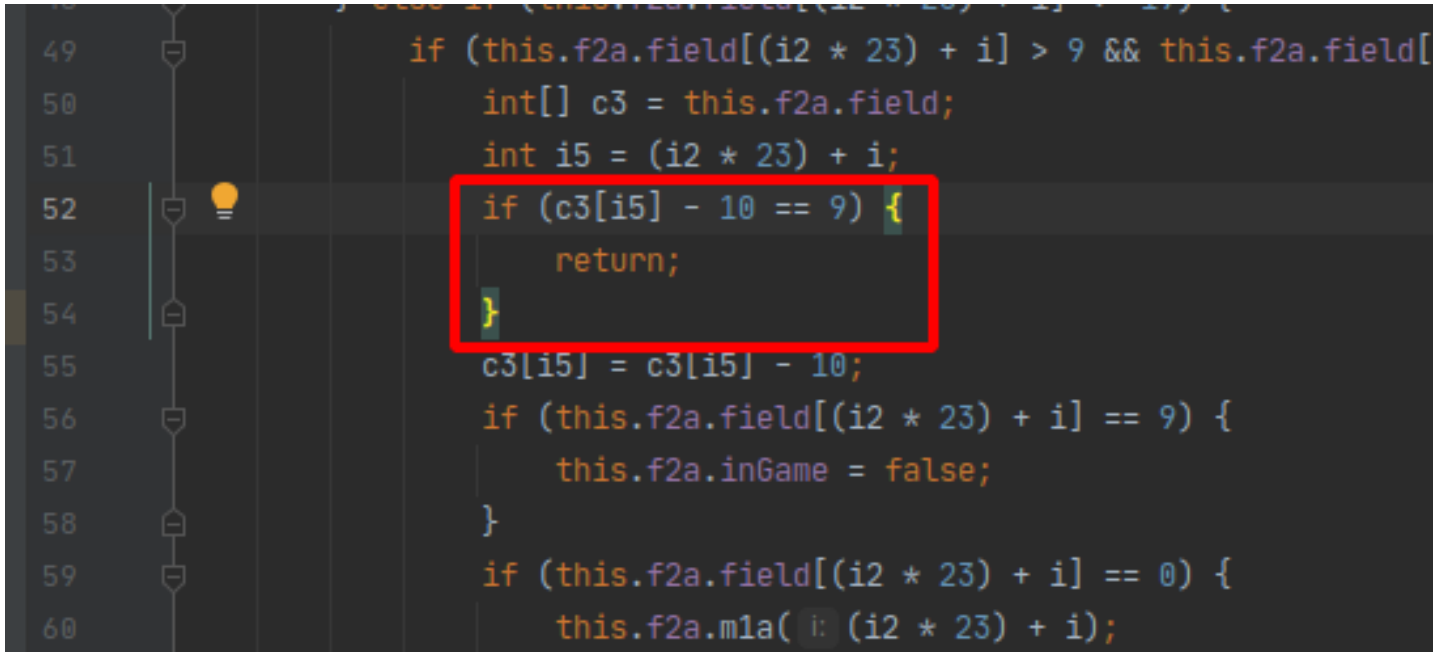
- **Solution:**

I used jadx to decompile it and I loaded the code in my java IDE

```
androldeed3br4:~/ctf/onapsis/files8kp/Lvl3$ ~/apps/jadx-1.1.0/bin/jadx --deobf --deobf-rewrite-cfg --deobf-use-sourcename --rename-flags all --no-replace-consts -d MineSweeper MineSweeper.jar
INFO - loading ...
INFO - processing ...
INFO - done
androldeed3br4:~/ctf/onapsis/files8kp/Lvl3$
```

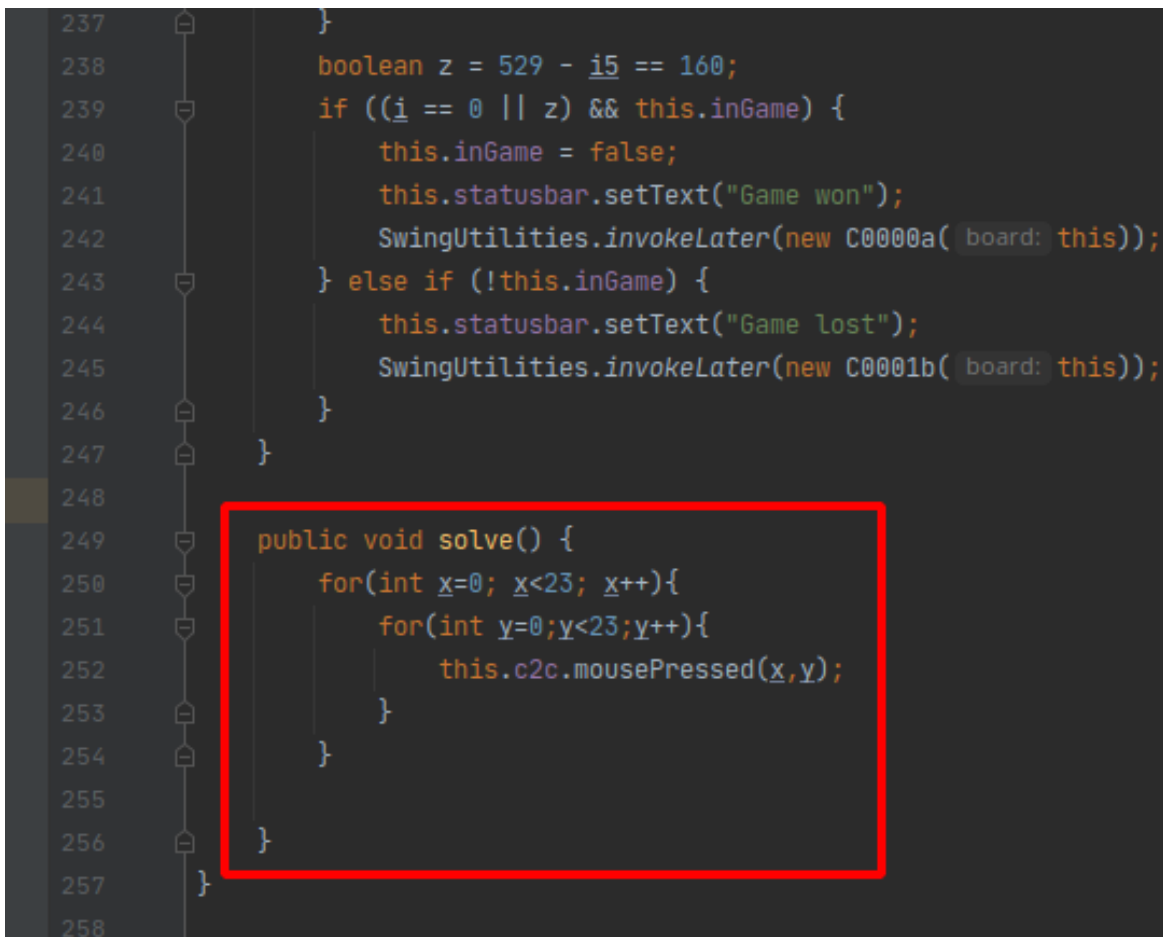
Basically, the modification done are:

1. When it is a mine, cut the flow and return.



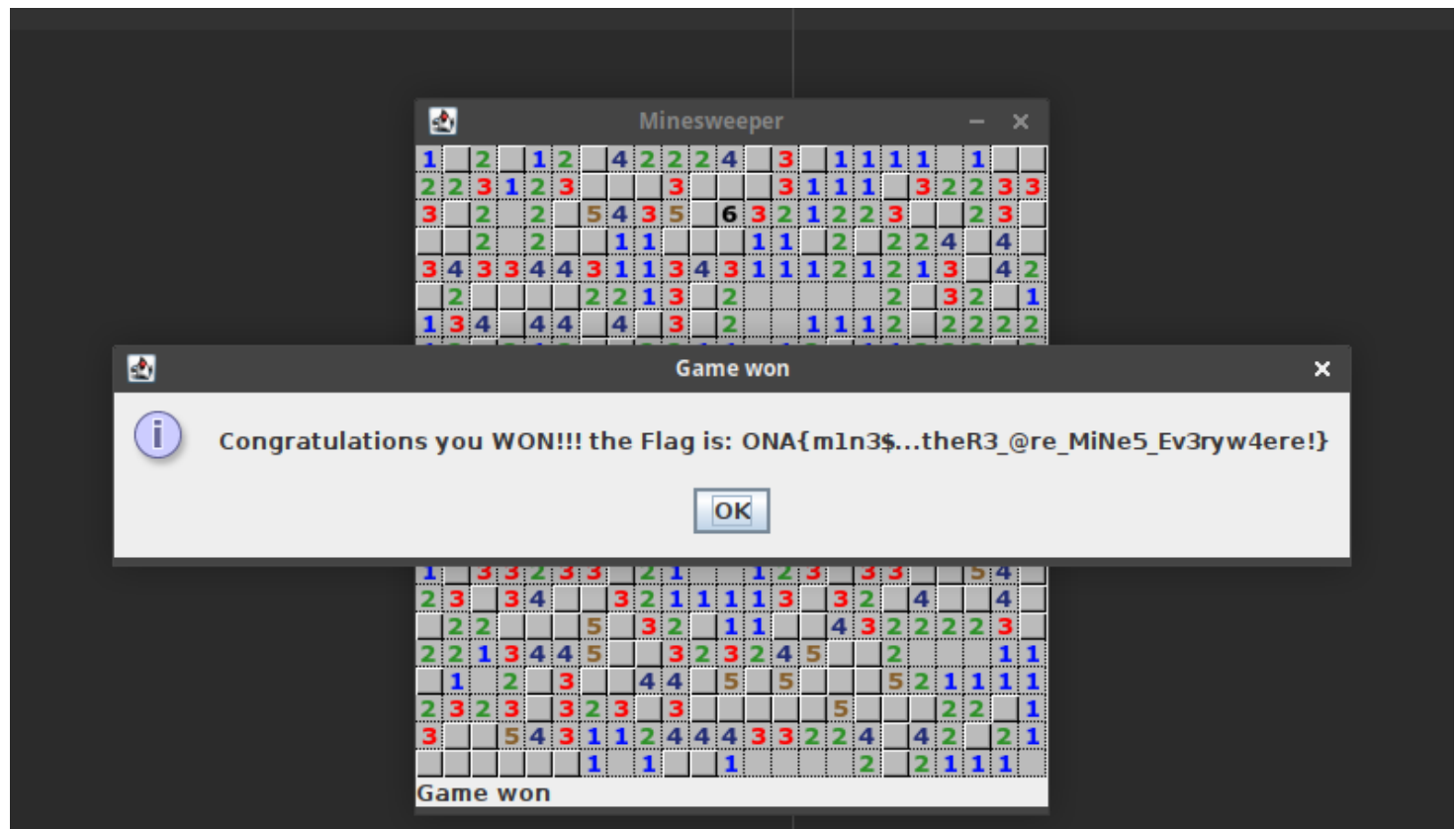
```
49         if (this.f2a.field[(i2 * 23) + i] > 9 && this.f2a.field[
50             int[] c3 = this.f2a.field;
51             int i5 = (i2 * 23) + i;
52             if (c3[i5] - 10 == 9) {
53                 return;
54             }
55             c3[i5] = c3[i5] - 10;
56             if (this.f2a.field[(i2 * 23) + i] == 9) {
57                 this.f2a.inGame = false;
58             }
59             if (this.f2a.field[(i2 * 23) + i] == 0) {
60                 this.f2a.m1a(i, (i2 * 23) + i);
```

2. Then, I created a method to go over all the board uncovering each cell.



```
237     }
238     boolean z = 529 - i5 == 160;
239     if ((i == 0 || z) && this.inGame) {
240         this.inGame = false;
241         this.statusbar.setText("Game won");
242         SwingUtilities.invokeLater(new C0000a( board: this));
243     } else if (!this.inGame) {
244         this.statusbar.setText("Game lost");
245         SwingUtilities.invokeLater(new C0001b( board: this));
246     }
247 }
248
249 public void solve() {
250     for(int x=0; x<23; x++){
251         for(int y=0; y<23; y++){
252             this.c2c.mousePressed(x, y);
253         }
254     }
255 }
256
257 }
258
```


Finally, as soon as I run the modified application it gives me the flag.



- **Gotten flag:**

ONA{m1n3\$...theR3_@re_MiNe5_Eve3ryw4ere!}

Level4

- **Precondition:**

previous flag: ONA{m1n3\$...theR3_@re_MiNe5_Ev3ryw4ere!}

- **Description:**

We built a maze so that you can not continue.

Exit this maze and you will find the flag you are looking for!

Developers say it's easy, but is harder when you can't see where you are going, and a monster is looking for you

- **Analysis:**

In 7z file there's only one ELF 64 bits binary. I run the executable and it showed a maze that, according the given description, has invisible walls.

In addition, I only have a fixed number of movements. Let's open the binary with a disassembler to take a look.

```
Where do you want to go now?:
Moving to the Right
You hit the electrified wall... Bad luck, try again!
androlde@d3br4:~/ctf/onapsis/filesBkp/Lvl4$
```

I realized that the executable has a flag to run in “dev” mode adding “-Debugg” as a parameter when the binary is executed in command line.

I also found a validation that avoids showing the flag if this is running in dev mode, so we are going to disable it.

• **Solution:**

The following changes were made in the binary:

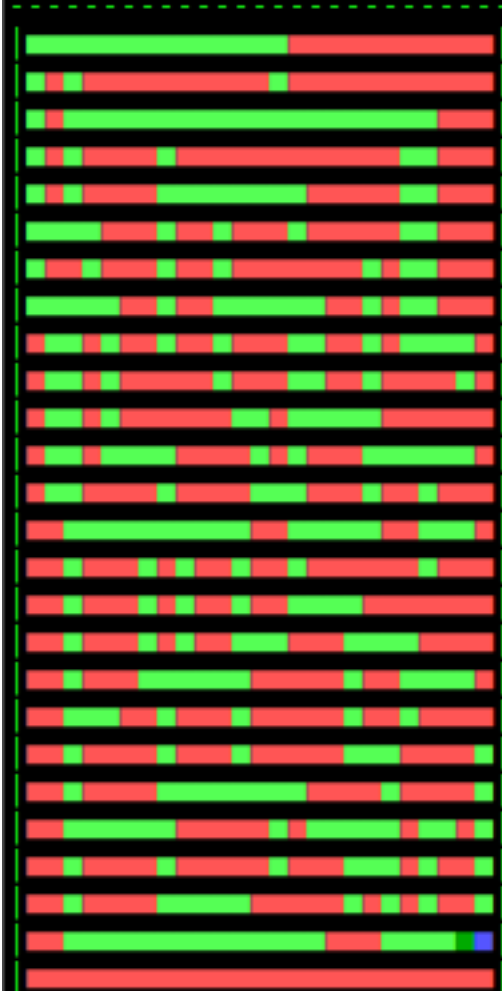
1. this enables the ability to include the “-Debugg” parameter or in command line
 - fileoffset: 1668
 - change: 0x46 to 0x54
2. this skips the validation that avoids showing the flag in dev mode
 - fileoffset: 12A0
 - change: 0x74 to 0xEB

After these changes, I executed the application passing it the mentioned parameter

```
$ ./MazeRuner -Debugg
```

That execution showed the maze but the walls are visible now. The last step is about to find, visually, as always, the right path and it will give me the flag

steps left: 1



Where do you want to go now?:

Moving to the Right

Great Job, your flag is: ONA{1_tr0pez0n_!=_ca1d4?}

androlde@d3br4:~/Downloads/drive-download-20200826T191558Z-001/Lvl4\$ █

- **Gotten flag:**

Gotten flag: ONA{1_tr0pez0n_!=_ca1d4?}

Level5

- **Precondition:**

previous flag: ONA{1_tr0pez0n!=_ca1d4?}

- **Description:**

Sudoku is my favorite game. You need to test your brain and look for all possibilities.

Of course, it is even harder if numbers are hexadecimal. Luckily, I'm able to save the game and continue whenever I want

- **Analysis:**

I started exploring the executable and, as the description said, It allows you to save the game and reload it again later.



The flag is built from some characters of the board so I just need to solve it, maybe with an sudoku's solver algorithm.

• Solution:

Checking the executable I found the way all data is stored in the file after saving the state of the game.

```
.text:0000000000401110      mov     edi, offset aSavingGame ; "Saving game"
.text:0000000000401115      mov     eax, 0
.text:000000000040111A      call    _printf
.text:000000000040111F      mov     edi, 2                ; seconds
.text:0000000000401124      call    _sleep
.text:0000000000401129      lea     rax, [rbp+filename]
.text:0000000000401130      mov     esi, offset aW        ; "w"
.text:0000000000401135      mov     rdi, rax              ; filename
.text:0000000000401138      call    _fopen
.text:000000000040113D      mov     [rbp+stream], rax
.text:0000000000401144      mov     eax, cs:L
.text:000000000040114A      xor     eax, 0FFFFFF80h
.text:000000000040114D      movsx   eax, al
.text:0000000000401150      mov     rdx, [rbp+stream]
.text:0000000000401157      mov     rsi, rdx              ; stream
.text:000000000040115A      mov     edi, eax              ; c
.text:000000000040115C      call    _fputc
.text:0000000000401161      mov     [rbp+var_A0], 0
.text:000000000040116B      jmp     short loc_4011C9
.text:000000000040116D      ; -----
.text:000000000040116D      loc_40116D:                ; CODE XREF: save+123+j
.text:000000000040116D      mov     [rbp+var_9C], 0
.text:0000000000401177      jmp     short loc_4011B9
.text:0000000000401179      ; -----
.text:0000000000401179      loc_401179:                ; CODE XREF: save+113+j
.text:0000000000401179      mov     eax, [rbp+var_9C]
.text:000000000040117F      cdq     edx
.text:0000000000401181      mov     edx, [rbp+var_A0]
.text:0000000000401187      movsxd  rdx, edx
.text:000000000040118A      shl     rdx, 4
.text:000000000040118E      add     rax, rdx
.text:0000000000401191      mov     eax, ds:PLAYER_BOARD[rax*4]
.text:0000000000401198      and     eax, 7Fh
.text:000000000040119B      xor     eax, 0FFFFFF80h
.text:000000000040119E      movsx   eax, al
.text:00000000004011A1      mov     rdx, [rbp+stream]
.text:00000000004011A8      mov     rsi, rdx              ; stream
.text:00000000004011AB      mov     edi, eax              ; c
.text:00000000004011AD      call    _fputc
.text:00000000004011B2      add     [rbp+var_9C], 1
```

I also found a web that can solve this type of hexa suduko game. The web is <https://www.dcode.fr/-hexadoku-sudoku-16-solver>

To read the obtained file, parse it and get the output that the web page needs, I wrote a python script that receives one parameters which is the path of the file obtained from the executable

```
import sys

def convert(letter):
    return ord(letter) ^ 0x80

def read_and_parse(input_file):
    with open(input_file, 'rb') as input_file:
        print("level: " + str(convert(input_file.read(1))))
        board = ""
        while 1:
```

```

byte = input_file.read(1)
if not byte:
    break
resulted_char = chr(convert(byte))
board += " " if resulted_char == '_' else resulted_char
print "[" + board + "]"

if __name__ == '__main__':
    read_and_parse(sys.argv[1])

```

First of all, I used the application feature to save the state of the board.

```

Modify a number: [SPACE].
Pause game: [p]

Chose a Name for the new game file:
unsolved_level0

```

The following image describes the execution of the script passing the file previously created as a parameter

```

androide@d3br4:~/ctf/onapsis/filesBkp/Lvl5$ python3 board_parser.py unsolved_level0
level: 0
[ 0 6f d e d1 7 4b b 9a1 32 0 6 4 ba7c15e 1 9 5 73 a39 f 54 2 4 d7 06 91 0 b1 ce dfac
f0 5e a7 1 4b 73 86 9 8 e3 1 ba7 a3 6 d f b1cf458 9 e 9 12 857 f 30 e a5 5 9 34 1 ]
androide@d3br4:~/ctf/onapsis/filesBkp/Lvl5$

```

The output indicates that the current level is 0. I copied the text within the brackets, pasted it in the board showed in the online tool (<https://www.dcode.fr/hexadoku-sudoku-16-solver>) and clicked "Solve Sudoku"



HEXADOKU (SUDOKU 16X16) SOLVER
Games and Solvers › Number Games › Hexadoku (Sudoku 16x16) Solver

Search for a tool

★ SEARCH A TOOL ON DCODE BY KEYWORDS:
e.g. type random GO

Results

4	7	C	0	B	6	F	2	D	9	5	1	3	E	A	8
3	D	1	A	7	0	8	5	F	C	E	4	B	9	2	6
B	E	8	5	C	9	A	1	7	6	3	2	F	4	D	0
F	6	2	9	4	D	E	3	0	8	B	A	7	C	1	5
E	2	A	C	F	1	6	4	9	D	0	B	5	8	7	3
6	8	D	7	A	3	9	B	E	F	1	5	4	0	C	2
5	4	3	F	D	7	2	0	6	A	C	8	9	1	B	E
0	9	B	1	8	5	C	E	4	7	2	3	6	D	F	A
C	F	0	2	6	B	4	8	5	E	9	D	A	7	3	1
D	1	4	B	2	A	5	7	3	0	8	6	E	F	9	C
8	5	9	E	3	F	1	C	2	B	A	7	D	6	0	4
A	3	7	6	9	E	0	D	1	4	F	C	2	5	8	B
1	C	F	4	5	8	D	A	B	2	6	9	0	3	E	7
9	A	E	D	1	2	3	6	8	5	7	0	C	B	4	F
2	B	6	3	0	4	7	F	C	1	D	E	8	A	5	9
7	0	5	8	E	C	B	9	A	3	4	F	1	2	6	D

HEXADOKU (SUDOKU 16X16) SOLVER
Games and Solvers › Number Games › Hexadoku (Sudoku 16x16) Solver

Summary

- Solve a Hexadoku grid
- How to fill the Hexadoku 16x16 grid?
- How does the step by step works?

Similar tools

- Sudoku Solver
- Sudozen (Sudoku 12x12) Solver
- Countdown Numbers Game
- Fill The Blanks Equation Solver
- Word's Value
- Magic Square
- Missing Numbers Calculator
- Mathador Solver
- Cryptarithm Solver
- Sum of Digits
- ★ All Tools ★

Support

- Paypal
- Patreon
- More

Forum/Help

DISCORD

Sponsored ads

Ads by Google

Sudoku 16X16 Free

Online Sudoku Solver

See also: [Sudoku Solver](#) — [Sudozen \(Sudoku 12x12\) Solver](#)

As you can see, I got the solution at the left side of the page. I just used a text editor to join all characters in one line in order to use it in another python script I created to write a file with the format that the sudoku application requires.

One line solution:

47C0B6F2D9513EA83D1A7085FCE4B926BE85C9A17632F4D0F6294DE308BA7C15E2ACF1649D0B58730

The other script I created receives three parameters: first, the level where we are; second, the filename of the file which will be created and third, the characters of the solution(in one line)

```
import sys

def convert(letter):
    return (hex(letter ^ 0x80)).upper()[2:]

def solve(p_level, p_outputfile, p_solution):
    with open(p_outputfile, 'wb') as output_file:
        hex_string = convert(p_level)
        for letter in p_solution:
            hex_string += convert(ord(letter))
        output_file.write(bytearray.fromhex(hex_string))
    print("done")
```



```
if __name__ == '__main__':
    level = int(sys.argv[1])
    outputfile = sys.argv[2]
    solution = sys.argv[3]
    solve(level, outputfile, solution)
```

```
androide@d3br4:~/ctf/onapsis/filesBkp/Lvl5$ python solution_parser.py 0 solved_level0 47C0B6F2D9513EA83D1A7085FCE4B926BE85C9A17632
F4D0F6294DE308BA7C15E2ACF1649D0B587368D7A39BEF1540C2543FD7206AC8918E09B185CE47236DFACF026B485E9DA731D14B2A573086EF9C859E3F1C2BA7D6
04A3769E0D14FC258B1CF458DAB26903E79AED12368570CB4F2B63047FC1DE8A597058ECB9A34F126D
done
androide@d3br4:~/ctf/onapsis/filesBkp/Lvl5$
```

After the execution, I went back to the application and loaded the “solved_level0” file.

```
47C0 B6F2 D951 3EA8
3D1A 7085 FCE4 B926
BE85 C9A1 7632 F4D0
F629 4DE3 08BA 7C15
-----
E2AC F164 9D0B 5873
68D7 A39B EF15 40C2
543F D720 6AC8 91BE
09B1 85CE 4723 6DFA
-----
CF02 6B48 5E9D A731
D14B 2A57 3086 EF9C
859E 3F1C 2BA7 D604
A376 9E0D 14FC 258B
-----
1CF4 58DA B269 03E7
9AED 1236 8570 CB4F
2B63 047F C1DE 8A59
7058 ECB9 A34F 126D
-----
Modify a number: [SPACE].
Pause game: [p]
Game Loaded Successfully! You are in level 0:

Well done, you solved level 1! Here you have a part of the Flag: Press SPACE to play level 2.
Flag (first part): ONA{sUd0_5uD0
Press [ENTER] to play level 2.
Hit Enter to start...█
```

We see that I got only the first part of the flag and it also says that there is a second level. The second level consists on the same process so I repeated exactly the same steps

```
-----  
A594|18F2|E06B|C73D|  
31FC|D4E6|7289|A50B|  
72DE|B9C0|35A4|1F86|  
8B06|73A5|1FCD|294E|  
-----  
D823|A071|F645|BCE9|  
C4A7|62BD|089E|315F|  
591F|CE38|AB27|06D4|  
E06B|5F49|CD31|8A72|  
-----  
6DC5|270B|84F3|9E1A|  
2F49|3C8E|D15A|60B7|  
1380|96DA|BE7C|42F5|  
BE7A|415F|2906|D8C3|  
-----  
4C52|8D67|931F|EBA0|  
9731|EA24|5CB0|FD68|  
FAE8|0B13|67D2|549C|  
06BD|F59C|4AE8|7321|  
-----  
Modify a number: [SPACE].  
Pause game: [p]  
Game Loaded Successfully! You are in level 1:  
  
Congratulations, you Win!!! Here is the rest of the Flag:  
  
Flag (final part): _5Ud0KUuu!!!}  
androlde@d3br4:~/ctf/onapsis/filesBkp/Lvl5$ █
```

- **Gotten flag:**

ONA{sUd0_5uDO_5Ud0KUuu!!!}

Level6

- **Precondition:**

previous flag: ONA{sUd0_5uDO_5Ud0KUuu!!!}

- **Description:**

Final level. You really are a true gamer! But this is not a simple game It is a Spies game.

Our Spy Agency released a new software to communicate with agents and assets. It uses encrypted messages to hide what we are saying.

We know that an enemy hacker recovered this app. He'd been playing around with it and was trying to recover the secret flag.

Can you find out if he succeeded?

Remember, the flag can be used to open the safe house, containing the final treasure!

- **Analysis:**

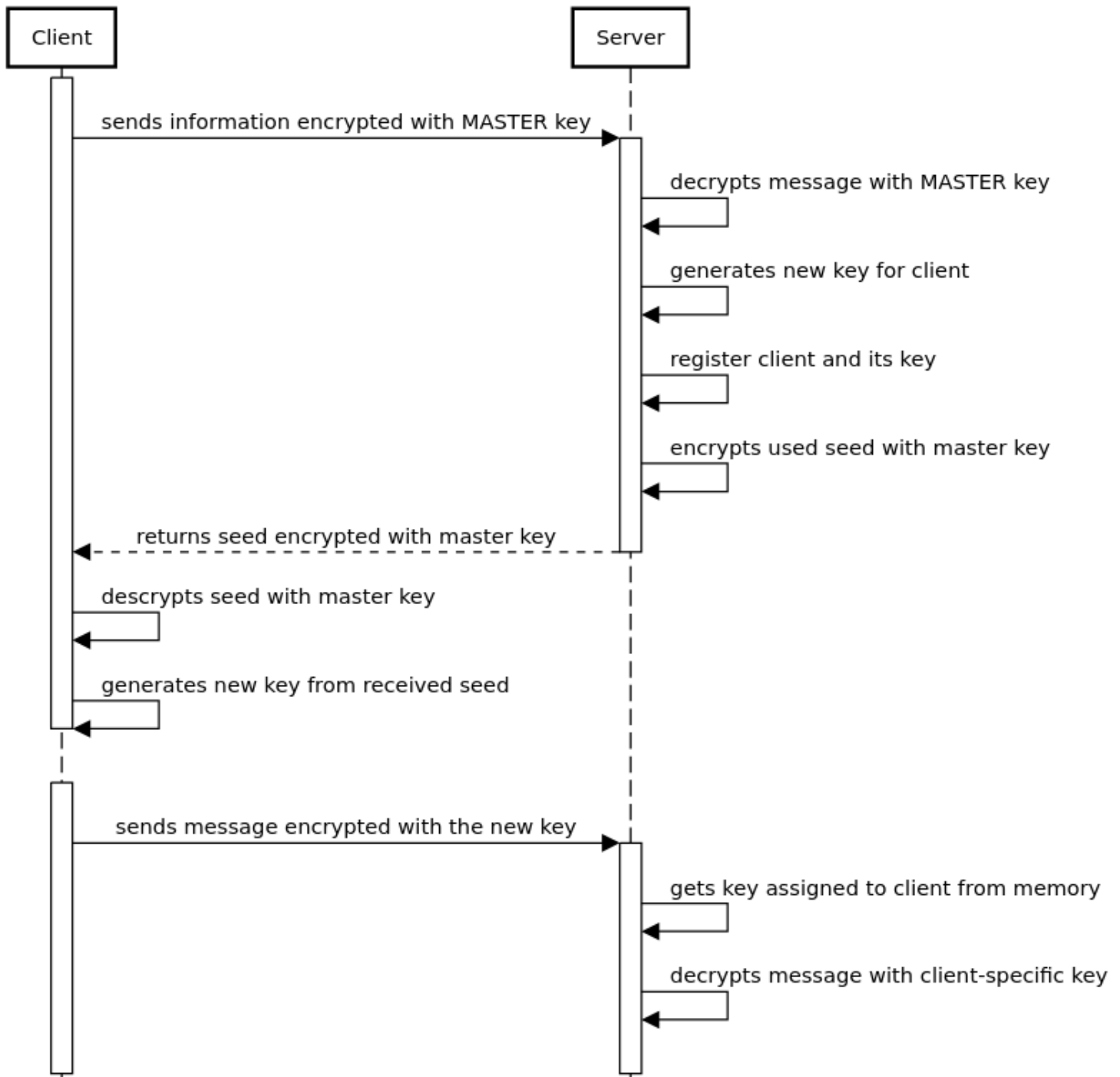
There are four files in the 7z file. Two jar files and two logs. The jar files are a Client and a Server application that allows "the agency", "agents" and "assets" have a secure way channel of communication between them.

The Client component has an option to retrieve a flag from the Server.

The goal is to analyse the way these two components exchange messages and decrypt all the messages stored(encrypted) in the logs files. I expect to find, in any of these messages, the flag of the challenge or a clue to a possible next step.

- Client and Server has a unique master key
- In every communication, the client(agent) starts sending its own information
- After the client's registration, this and the server use the symetric key created to "secure" the communication

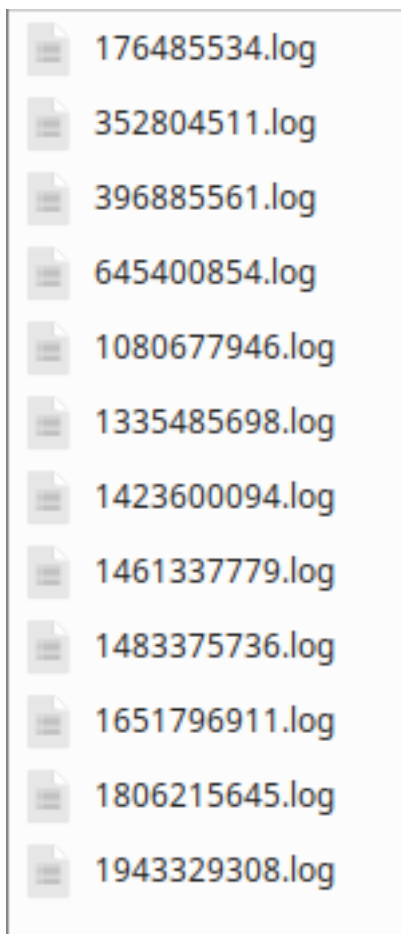
Register of client and simple message send



IMPORTANT: The client has the ability to generate a new key. For this, client generates a new “random” key that will be sent encrypted with the previous key.

- **Solution:**

Every agent has an “id” so I divided the log of each “agent” taking this id as reference.



First, I decided to analyse only the agents that received the flag from the server.

```
androlde@d3br4:~/ctf/onapsis/filesBkp/Lvl6/communications$ grep -R "Flag sent to secret agent" .
./645400854.log:Flag sent to secret agent -645400854.
./1461337779.log:Flag sent to secret agent -1461337779.
./1943329308.log:Flag sent to secret agent 1943329308.
androlde@d3br4:~/ctf/onapsis/filesBkp/Lvl6/communications$
```

For that task, I used the class `crypto.Key` included in both jars and created the following main class in java to decrypt all the messages for agent with id "-1461337779"

```
package solution;

import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.util.logging.Logger;

public class Main {
    private static final Logger logger;
    private Key currentKey;

    static {
        System.setProperty("java.util.logging.SimpleFormatter.format", "%1$tF %1$tT %2$s [%4$s] %5$s %n");
        logger = Logger.getLogger(Main.class.getName());
    }

    public static void main(String[] args) {
        Main main = new Main();
        String[] arr1461337779 = new String[]{"wqZUwpECES0Aw5nCm1E=", "wq5Zwp8DH80Cw5PCn1jClcK1S8Ki",
            "CckBwqfDkiEewofDog==", "CsKFwrfDkzMF", "CsKFwrfDkzMFwqXDpznCss0fH80Ew7w=",
            "CsKFwrfDkzMFwqXDpznCss0fH80Ew7zCv35v", "CsKFwrfDkzMFwqXDpznCss0fH80Ew7zCvX5t",
            "CsKVwqTDhCQhwpTDtTnCts0HAs0S", "CckBwqfDkiEewofDohnCtM0YFc0E",
            "CsKVwqTDhCQiwqDDpTjCpM0cIM0Xw6vDvzsxAjw=", "CsKVwqTDhCQiwqDDpTjCpM0cIM0Xw6vDvzsxAjwkw4Z/",
            "CsKVwqTDhCQuwqbDoynCs80NBMOpw4jDrT8tBzdnwpB9w4nDpA==",
            "CsKVwqTDhCQuwqbDoynCs80NBMOpw4jDrT8tBzdnwpATw4rDpcK0",
            "CsKVwqTDhCQuwqbDoynCs80NBMOpw4jDrT8tBzdnwpATw4rDpcK0VA=="
        }
```

```

        "CsKVwqTDhCQuwqbDoynCs80NBMOpw4jDrT8tBzdnwpATw4rDpcKOVGw=",
        "CsKVwqTDhCQuwqbDoynCs80NBMOpw4jDrT8tBzdnwpATw4rDpcKOVGzDlG==",
        "CsKVwqTDhCQiwqDDpTjCpM0cIMOXw6vDvzsxAjwkW4Z/w4/Dog==",
        "CsKVwqTDhCQuwqbDoynCs80NBMOpw4jDrT8tBzdnwpATw4rDpcK0HA==",
        "QMOZw6PDhWZGw4DCvnzDucKcQ8KBw7vDqCk/EjoJw41+w4LDocKEAjJdkc0sw4BgQg==",
        "aMONw6XCjGRcw4bCq3/DrMKQXcKHwqvCoX5vXWshw5l5w47DusKFWQ==",
        "WENXQkBAUllcQlZTVF1KXlHew0hTRVlIWUpUU0FfXVtaSVBTXEHfUKReQlNSVkJAQVNWxELZT0BbXkBEWkM=",
        "QlIUtlTlSLNZQlTMrvpCXA==",
        "WENXQkBAUllcQlZTVF1KXlHew0hTUF5CSFhDWUhrTExBsLBDTERfUg==",
        "XUJJeRkxLXF9LXVdZVF1ZW1BBTVVKUftZS0dFQk9PV1NBSFZfTURF",
        "WENXQkBAUllcQlZTVF1KXlHew0hTV1lMw1JPRVBZTEVSTL5ISl4=", "RkVYWdTW0xBU0xCXg==",
        "X0hUSUxeUktQU1BCREBITFBLR1FfS1hES0pYQl8=",
        "QkxXSExFXfHgSEZEQk1MRk9dUlhIR1hCREdIQkRTRURfQUtDXUhVTVlaVFNWExSSVNfVL5JT0FHTlg=",
        "RUVUU0FIUF9QUZLXVJKTfHeTVtDVE5Q",
        "VKJeSUXPSkhJU1lDVU1KXV5JTVhYVL5RRkBeWE9aW0FfU1tIXUlpWll0TVE=", "cG56"};
    main.solve(arr1461337779);
}

public static boolean isFromCharset(String stringToEncode, Charset charset) {
    return charset.newEncoder().canEncode(stringToEncode);
}

private void solve(String[] messages) {
    currentKey = Key.SERVER_KEY;
    for (int i = 0; i < messages.length; i++) {
        String message = decryptAndVerify(i, messages);
        logger.info(message);
        if (i == 1) {
            currentKey = new Key(Long.parseLong(message));
        }
    }
}

private String decryptAndVerify(int i, String[] messages) {
    String decryptedMessage = currentKey.decrypt(messages[i]);
    Key tmpKey = currentKey;
    if (!isFromCharset(decryptedMessage, StandardCharsets.US_ASCII)) {
        tmpKey = new Key(currentKey.decrypt(messages[i - 1]));
        decryptedMessage = tmpKey.decrypt(messages[i]);
    }
    if (isFromCharset(decryptedMessage, StandardCharsets.US_ASCII)) {
        currentKey = tmpKey;
    } else {
        currentKey = new Key(currentKey.decrypt(messages[i]));
    }
    return decryptedMessage;
}
}

```

After running the created java class, at the end of the decrypted communication, We can see the flag of the challenge.

```
2020-08-30 21:42:05 solution.Main solve [INFO] 987654321
2020-08-30 21:42:05 solution.Main solve [INFO] 1597869788444
2020-08-30 21:42:05 solution.Main solve [INFO] Password
2020-08-30 21:42:05 solution.Main solve [INFO] Secret
2020-08-30 21:42:05 solution.Main solve [INFO] SecretPassword
2020-08-30 21:42:05 solution.Main solve [INFO] SecretPassword321
2020-08-30 21:42:05 solution.Main solve [INFO] SecretPassword123
2020-08-30 21:42:05 solution.Main solve [INFO] SuperPassword
2020-08-30 21:42:05 solution.Main solve [INFO] PasswordSuper
2020-08-30 21:42:05 solution.Main solve [INFO] SuperSecretPassword
2020-08-30 21:42:05 solution.Main solve [INFO] SuperSecretPassword123
2020-08-30 21:42:05 solution.Main solve [INFO] Super_Secret_Password123
2020-08-30 21:42:05 solution.Main solve [INFO] Super_Secret_Password_123
2020-08-30 21:42:05 solution.Main solve [INFO] Super_Secret_Password_1234
2020-08-30 21:42:05 solution.Main solve [INFO] Super_Secret_Password_12345
2020-08-30 21:42:05 solution.Main solve [INFO] Super_Secret_Password_123456
2020-08-30 21:42:05 solution.Main solve [INFO] SuperSecretPassword12345
2020-08-30 21:42:05 solution.Main solve [INFO] Super_Secret_Password_123.
2020-08-30 21:42:05 solution.Main solve [INFO] a97d075868437cdeabb692969ba18a63
2020-08-30 21:42:05 solution.Main solve [INFO] 1-1-2-3-5-8-13-21-34-55-89
2020-08-30 21:42:05 solution.Main solve [INFO] information english. please provide details about next mission
2020-08-30 21:42:05 solution.Main solve [INFO] specify location
2020-08-30 21:42:05 solution.Main solve [INFO] information english. ekoparty. argentina
2020-08-30 21:42:05 solution.Main solve [INFO] look for potential agents to be recruit
2020-08-30 21:42:05 solution.Main solve [INFO] information english. black hat hackers
2020-08-30 21:42:05 solution.Main solve [INFO] white hat too
2020-08-30 21:42:05 solution.Main solve [INFO] need safe house after mission
2020-08-30 21:42:05 solution.Main solve [INFO] safe house is at palermo. you will need the flag. good luck
2020-08-30 21:42:05 solution.Main solve [INFO] the secret flag is ona{}
2020-08-30 21:42:05 solution.Main solve [INFO] the secret flag is ona{hackers.love.randoms}
2020-08-30 21:42:05 solution.Main solve [INFO] good bye. and good luck. you will need it...
2020-08-30 21:42:05 solution.Main solve [INFO] ACK
```

- **Gotten flag:**

ona{hackers.love.randoms}