[CTF Series #1] The Reverse Engineering Challenge

September 7th 2020

@GhouLSec
GhouL
Originate from medium @cyjien

Objective:

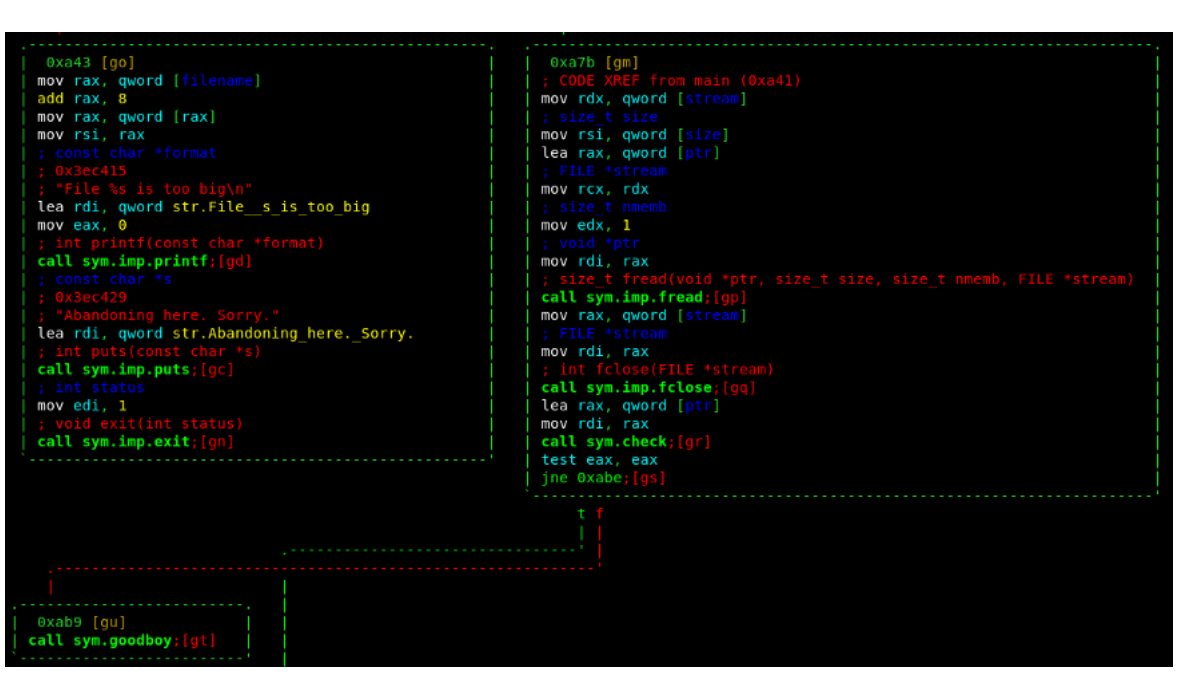
To get the flag from the binary (ELF) file.

Topics Covered:

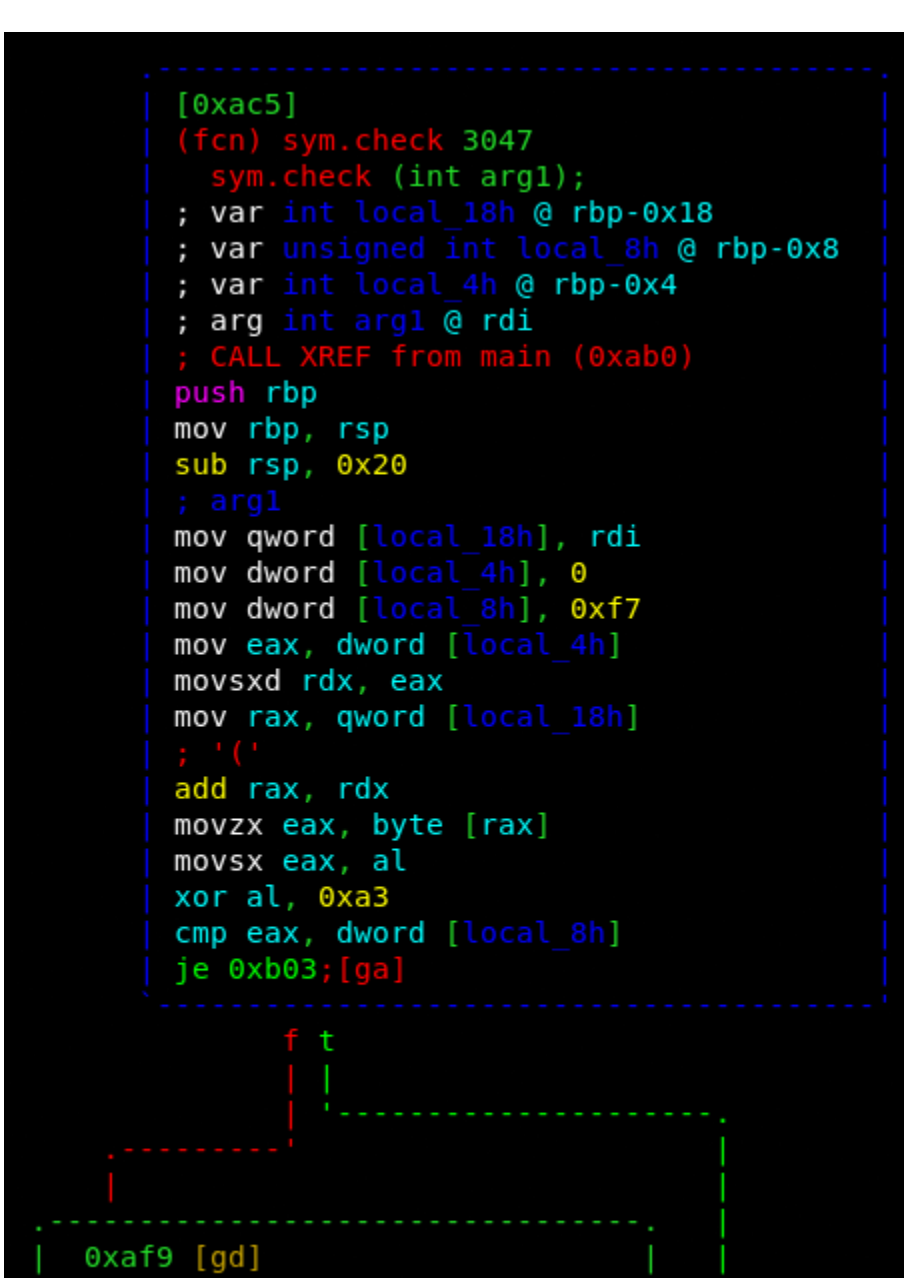
1. Radare2, x32/x64 dbg
2. Linux Command (objdump, awk, cut and grep)
3. Python Scripting

Procedure:

Here are the ideas on how to solve this challenge :D. Let's perform static analysis on the binary file by using radare2 in linux machine (my favourite debugging tools).



After glancing through the assembly codes, the binary looks like it will receive a file as a parameter and read it. It has a check and goodboy function looks that looks suspicious that will need further investigation on it.

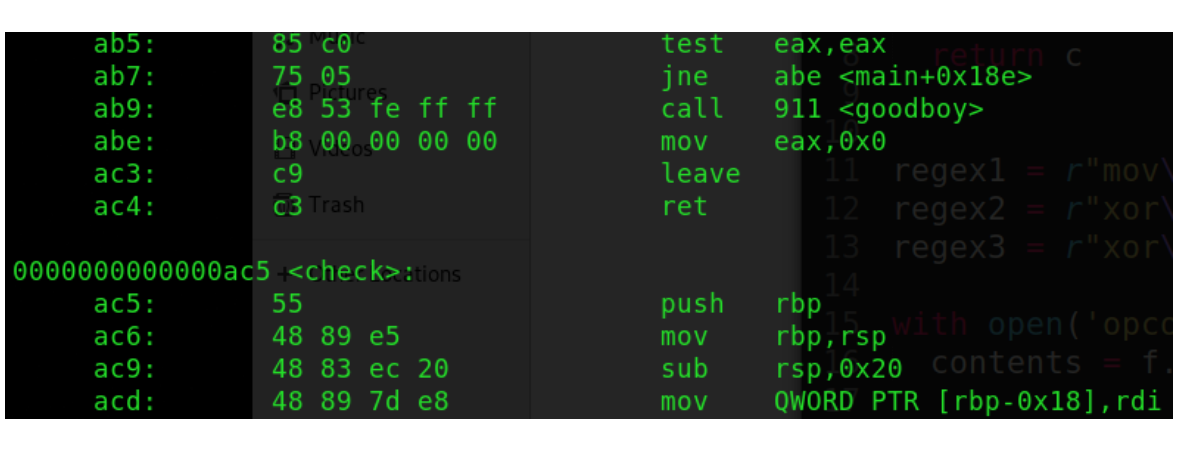


The code snippet on the left is the check function. It was very obvious that the value of eax register will compare with the value in the [**local_8h**] also known as **ebp-0x08h** to continue with its process.

Then, try to look upwards to understand where does the value of eax and [**local_8h**] comes from. There is a xor operation on the al (the lower bytes of the eax) with a constant value. If the compared value is not the same, it will goes to **badboy** function in which the program will terminated :(

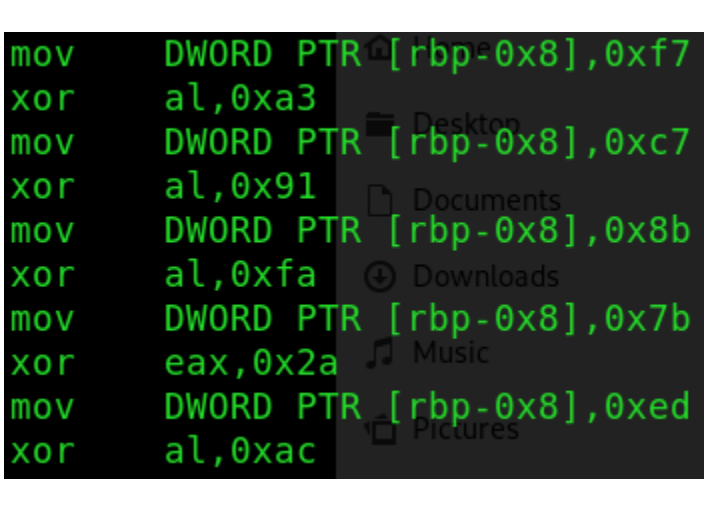
So, we just need to get the value of the eax that equals to the compared value to prevent the code run the badboy function. Since the xor operation is reversible, then we can get the correct eax value by xor the **constant** with the **[local_8h]** (e.g 0xf7 ^ 0xa3). However, there are too many blocks of code that the be xor. So, it cannot be done manually, therefore a script is needed to make our life easier. But before to write the script, we need to extract and filter all the unnecessary opcode. Objdump will help us here.

```
objdump -d -M intel ch30.bin
```



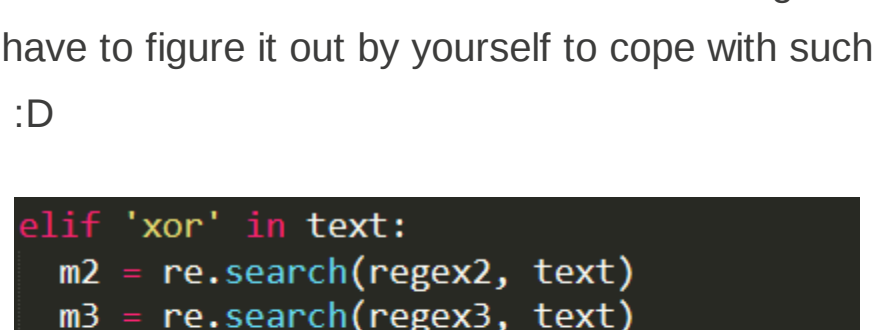
Hmm. The results are very long and we need to do some filter on it. Here, i will use the linux string manipulator command: **awk**, **cut** and **grep**. I will leave the command used here and i wont explain it in detail. The command here was mixed with the command in the available writeup (this only could be accessed after challenge was solved...oops…thus some of the the **awk** and **cut** command are redacted with 'x' character).

```
objdump -d -M intel xxx | awk -F'xx' -v RS="xxxx" 'xxxxxxxxxx' | cut -fx | grep "some mov and xor"
```



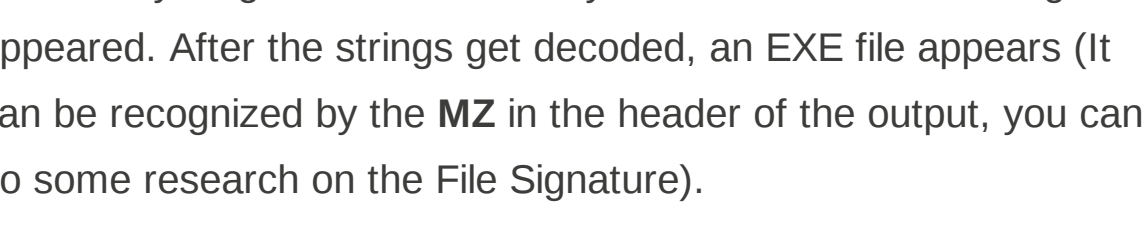
Here are the important opcode needed to solve this challenge. In order to get the hex code inside these large piece of opcode, i wrote a python script to automate all the process. Here is the code snippet.

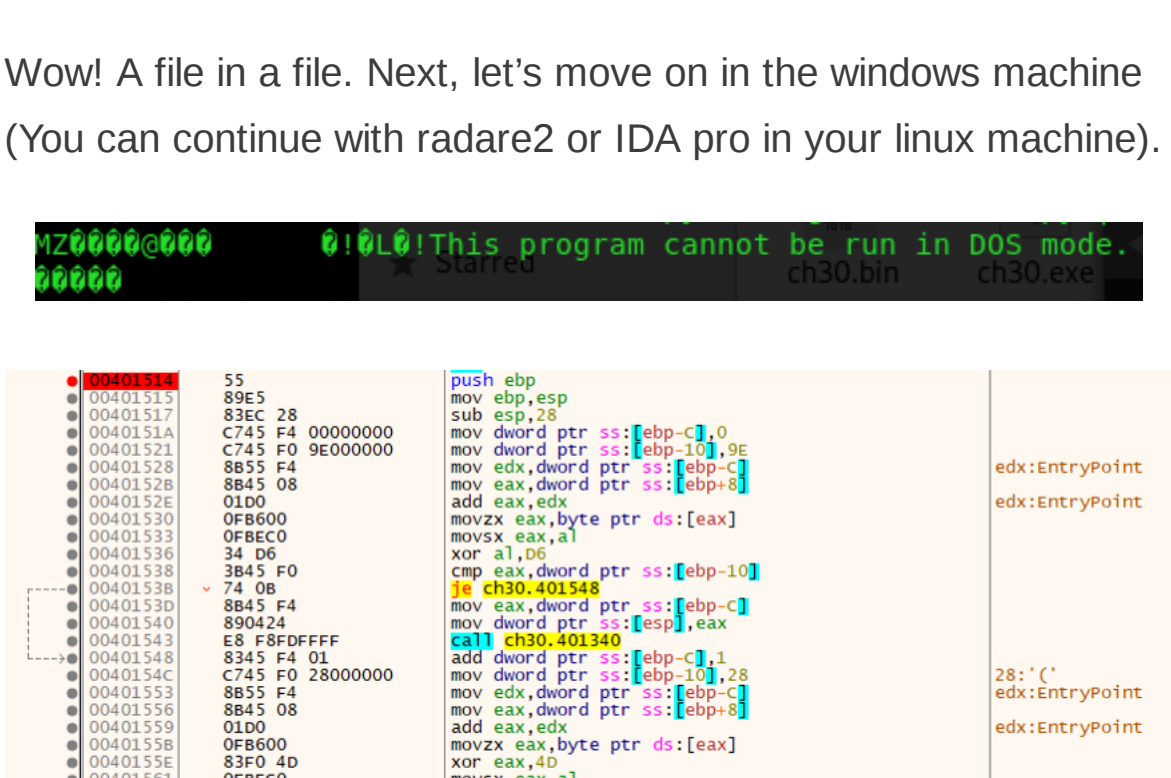
Basically, the idea is to using **regex** and **conditional operation** to filter out all the unnecessary strings then xor them to get the flag. But somehow there are some value that didn't get xor at all. So, you have to figure it out by yourself to cope with such a situation :D



After everything was done correctly, a base64 encoded strings appeared. After the strings get decoded, an EXE file appears (It can be recognized by the **MZ** in the header of the output, you can do some research on the File Signature).

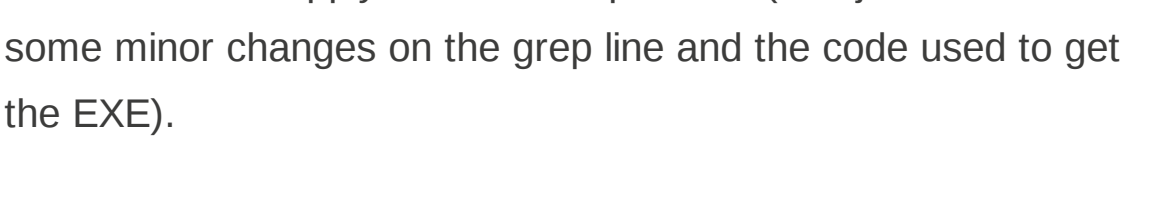
Wow! A file in a file. Next, let's move on in the windows machine (You can continue with radare2 or IDA pro in your linux machine).





From the code snippet, it can be seen that the overall function of the EXE file is same with the previous ELF file. So, the same solution can be apply to solve the problem (You just need do some minor changes on the grep line and the code used to get the EXE).

Finally! The flag comes out as i expected.



That's all for the write up, I hope you guys did enjoy my first ever write up on a reverse engineering challenge. Cheers! I'm also hoping that i can continue to publish some write up for the interesting challenges in the future. Oh ya, forgot to tell you guys this challenge is c-r-a-c-k-m-e a-u-t-o-m-a-t-i-n-g.

Previously published at https://medium.com/@cyjien/my-first-ever-write-up-on-the-reverse-engineering-challenge-6077b81b3021?source=friends_link&sk=a72d8e00069e5178cbc58370109d41b7

@GhouLSec
GhouL
Memeislife