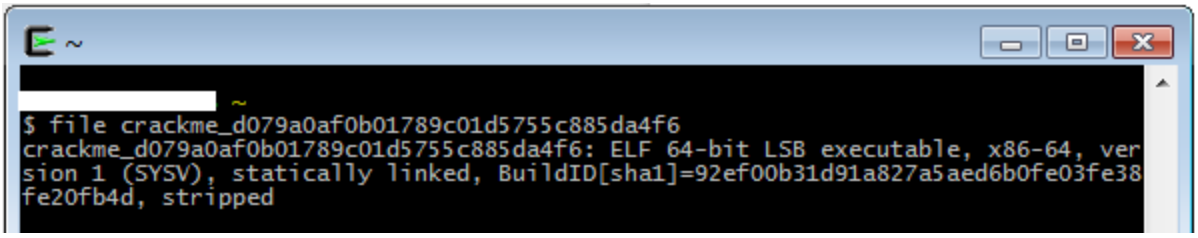
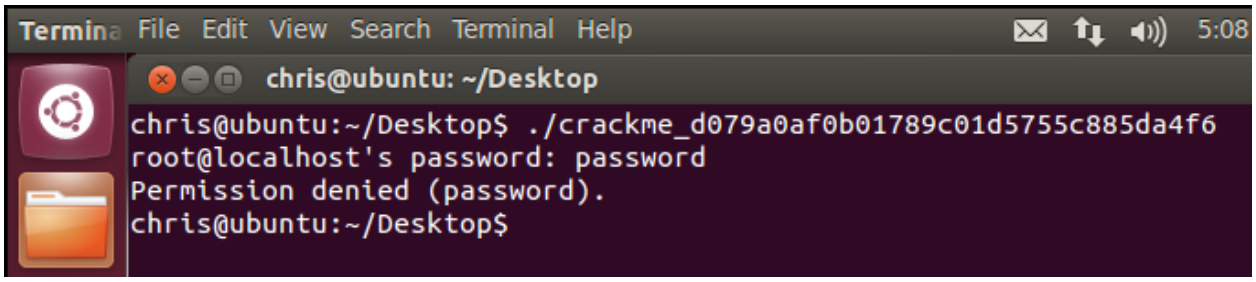
Codegate 2014オンライン予選Write Up
dodoCrackme (Reversing) 200

作成：田中クリス (Yu-Lu Chris Liu)
和訳：田中ザック

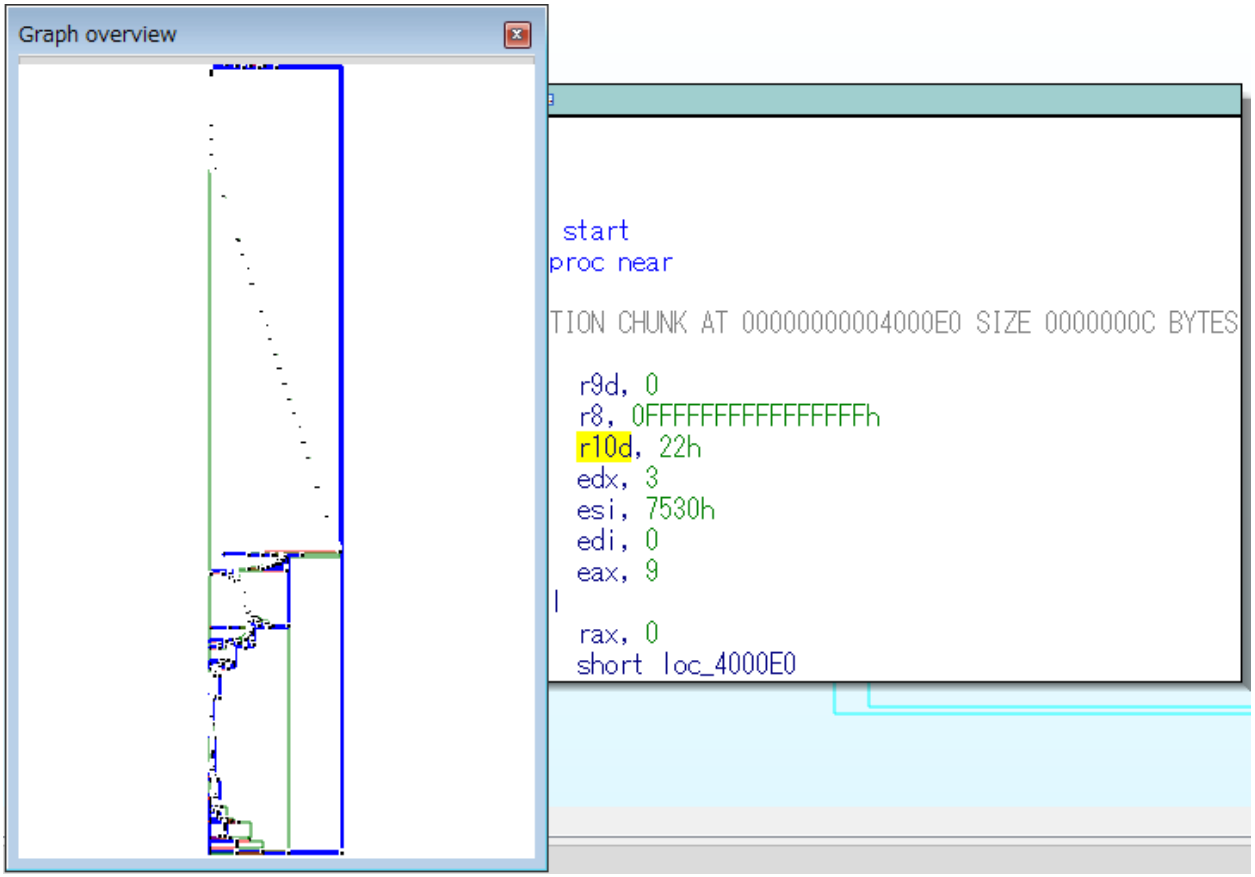
First I downloaded the file, and instantly did a file command on it to see what environment it will need to be executed. まず、ファイルをダウンロードしてfileで特定しました。



The result yields in a ELF 64-bit LSB executable. So next I fire up my Ubuntu 12.04 64-bit running on VMWare, and executed the file. ELF 64ビットのLSB実行ファイルだったので、Ubuntu 12.04 64ビットのVMで実行してみました。



At first I was surprised because it asks the password for root, but since this is a crackme, I assumed this is what we have to bypass. ルートのパスワードを聞かれるので驚いたが、crackmeなのでこれを回避しないといけないと思いました。Next, as everybody else would do I loaded the executable into IDA pro, trying to figure out the execution flows and important functions. 次、IDA proで開いて実行の流れと重要な関数を見てみました。But then IDA's graph showed that this program has been deliberately coded so that no normal human can benefit from IDA. At least I gave up right after seeing it. 残念ながらリバース・エンジニアリング対策でコードが難読化されたから、見た瞬間で諦めました。

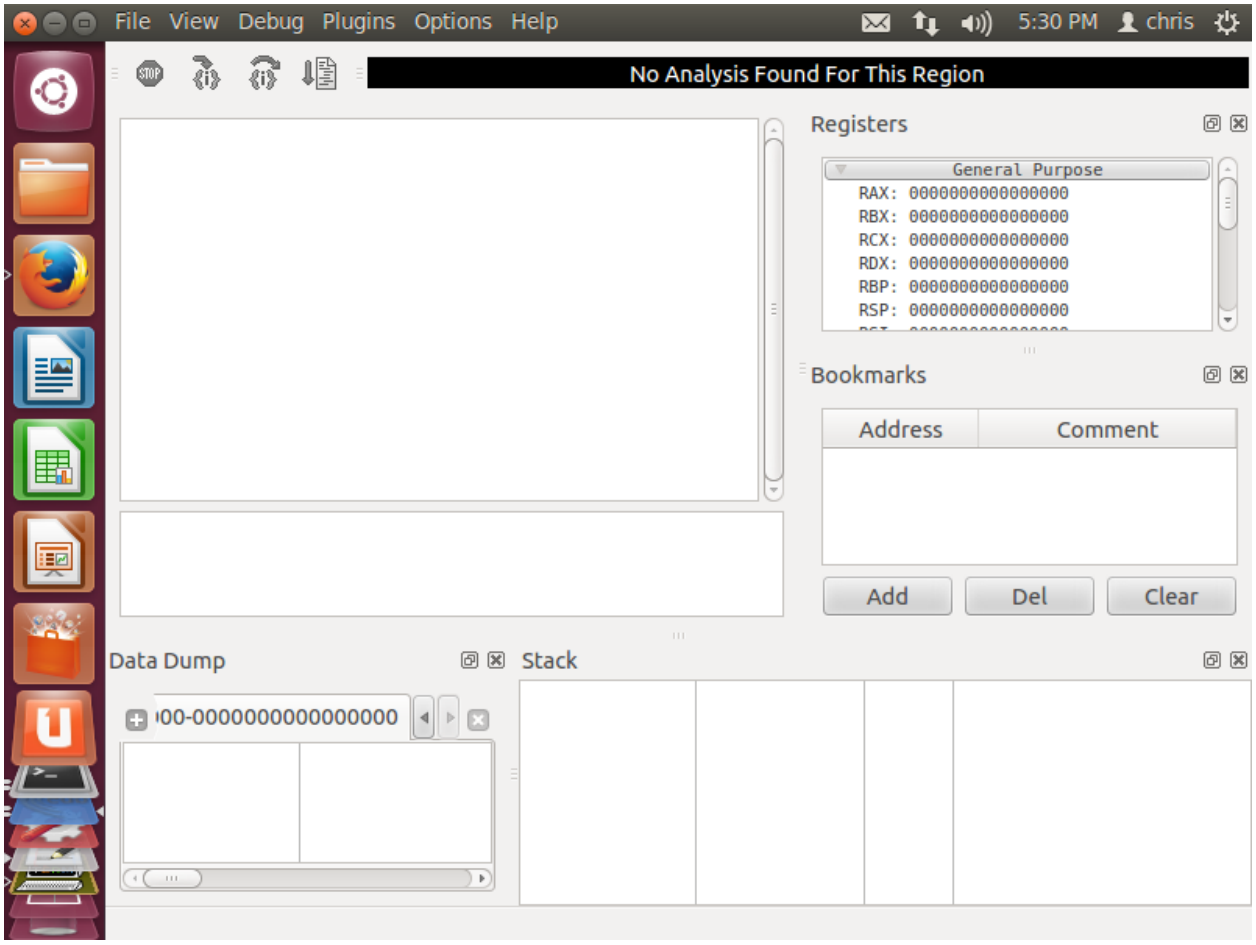


So my next hope is to run the executable in a debugger and see if I can better understand the logic inside this crackme. ということでデバッガーで見ることにしました。 I choose to use edb (evan's debugger) since I don't want to pull my my cheatsheet while using gdb inside linux. GDBのコマンドを忘れたからEDB(EvanさんのGUIデバッガー）を使いました。

The intstallation of edb on Ubuntu is quite simple. インストールは簡単

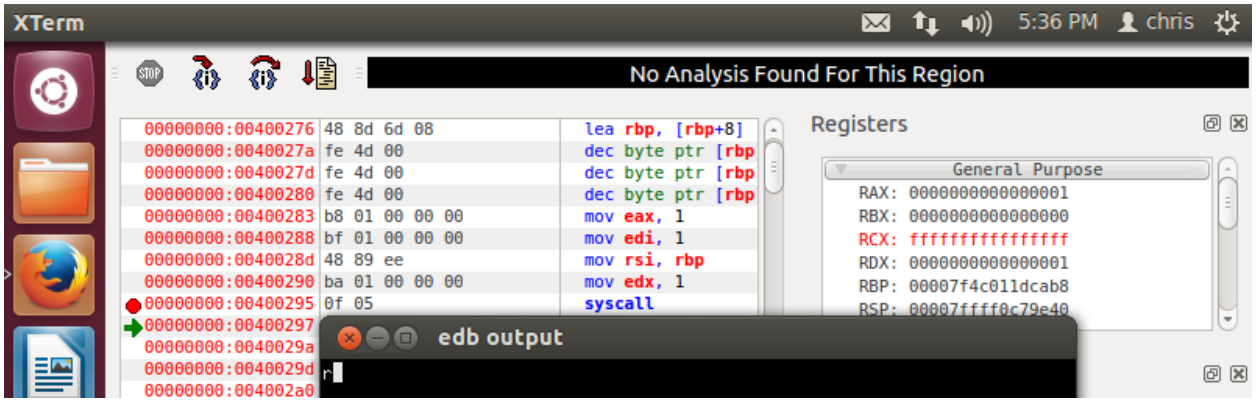
1. Download edb from http://freecode.com/projects/edebugger
1. http://freecode.com/projects/edebuggerからダウンロード。
2. In Ubuntu, type "sudo apt-get install libqt4-dev libboost-all-dev", this may take a while
2. 時間かかるけど「sudo apt-get install libqt4-dev libboost-all-dev」
3. Untar the file downloaded, go into the folder
3. untarして、フォルダーにcd
4. Type: "qmake"
5. Type: "make"
6. Start edb with the edb executable after you successfully compiled it
6. edbを実行すると

This is how edb looks like:こんな感じ



It kind of looks like OllyDbg on Windows, although it lacks a lot of the functions in Olly. WindowsのOllyDbgと似ていますが、Ollyほどの機能が付いていません。But this won't afect our analysis. しかし、この問題には十分です。All the hotkeys are the same, F7 for step through, F8 for step over, F9 for execute, and etc. ホットキーは同じ、F7→Step Through、F8→Step Over、F9→実行、、

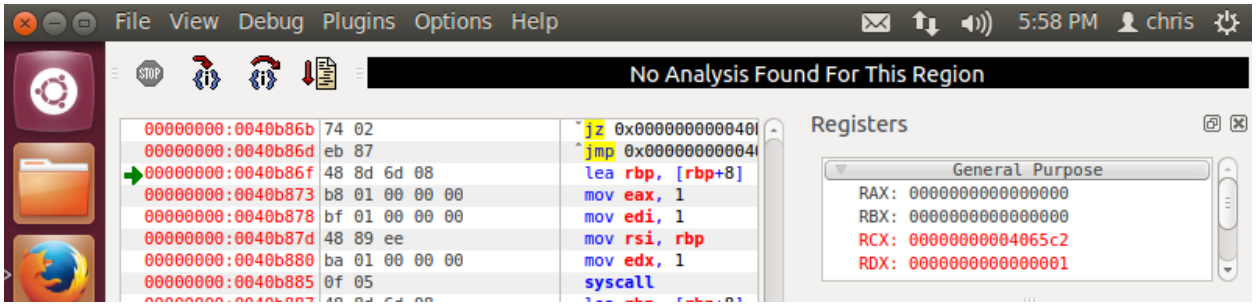
So I stepped through until I got to a "syscall" at 0x00400295 and the letter "r" was printed on the terminal. 命令をステップして0x00400295のsyscallに辿り着いたらターミナルに「ｒ」が表示されました。If you continue, you will see that every character in "root@localhost's password:" which gets shown when executing the binary are all printed from making syscall. ステップをし続けると、「root@localhost's password:」の全ての文字が同じようにこのsyscallで表示されます。In fact most (or all) or the printings to our terminals are made from syscall inside this binary. 実はこのバイナリが何かを表示する時にこのsysycallを使います。



So being a CTF layman myself, I guessed that the flag we are looking for should be a md5 hash (which later turned out to be wrong), so we should be looking at syscall inside a loop, or 32 syscalls that are close to each other. まだCTFの素人なのでフラグがMD5のハッシュかなと思って、取り敢えず32回繰り返されるループの中のsyscall若しくは32個のsyscallが並んでいる所を探しました。（MD5ハッシュは32文字だから）（後から分かったが、フラグは実はMDハッシュではなかった）To get a better picture I dumped the assemblies by objdump, and specifically focusing on syscalls. バイナリを把握するためにobjdumpを使ってアセンブリをダンプして、syscallを探しました。

```
chris@ubuntu:~/Desktop$ objdump -d crackme_d079a0af0b01789c01d5755c885da4f6 | gr
ep syscall
  4000ea:       0f 05                   syscall
  400113:       0f 05                   syscall
  400150:       0f 05                   syscall
  400295:       0f 05                   syscall
  4002b2:       0f 05                   syscall
  4002c6:       0f 05                   syscall
```

Unfortunately I could not find 32 syscalls but I did find 30 syscalls starting from 0x40b885 that are closed to each other. 残念ながら32個のsyscallが並んでいる所はなかったけど、アドレス0x40b885から30個のsyscallはありました。So I went to that address to poke around. 従って、そのアドレスに飛んで色々見てみました。Since all the characters referred from using a pointer stored in the rbp, I decide to place a breakpoint on address 0x40b86f to check out what this could be pointing to. 全ての文字がRBPにあるポインタを参照していたので、0x40b86fにもブレイクポイントを付けました。



I ran the program, input the password (just whatever) and sure enough the program stops at 0x40b86f. プログラムを実行して、適当にパスワードを入れたら、無事に0x40b86fのブレイクポイントに止まりました。Then I go to the address pointed by rbp, 0x7f09ba96eab0 in my machine, and see what this memory address holds. RBPにポイントされているアドレスの0x7f09ba96eab0のメモリーを見てみると、フラグが書かれていました！And I guess I am just lucky, because close to this location, there's our flag. It turns out that it is not a 32 byte string after all…. 結局32バイトのMD5ハッシュではかった。

Data Dump

00007f09ba96b000-00007f09ba973000

```
00007f09:ba96eb20 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00007f09:ba96eb30 d6 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 Ö...............
00007f09:ba96eb40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00007f09:ba96eb50 00 00 00 00 00 00 00 00 48 00 00 00 00 00 00 00 ........H.......
00007f09:ba96eb60 00 00 00 00 00 00 00 00 34 00 00 00 00 00 00 00 ........4.......
00007f09:ba96eb70 00 00 00 00 00 00 00 00 50 00 00 00 00 00 00 00 ........P.......
00007f09:ba96eb80 00 00 00 00 00 00 00 00 50 00 00 00 00 00 00 00 ........P.......
00007f09:ba96eb90 00 00 00 00 00 00 00 00 59 00 00 00 00 00 00 00 ........Y.......
00007f09:ba96eba0 00 00 00 00 00 00 00 00 5f 00 00 00 00 00 00 00 ........_.......
00007f09:ba96ebb0 00 00 00 00 00 00 00 00 43 00 00 00 00 00 00 00 ........C.......
00007f09:ba96ebc0 00 00 00 00 00 00 00 00 30 00 00 00 00 00 00 00 ........0.......
00007f09:ba96ebd0 00 00 00 00 00 00 00 00 44 00 00 00 00 00 00 00 ........D.......
00007f09:ba96ebe0 00 00 00 00 00 00 00 00 45 00 00 00 00 00 00 00 ........E.......
00007f09:ba96ebf0 00 00 00 00 00 00 00 00 47 00 00 00 00 00 00 00 ........G.......
00007f09:ba96ec00 00 00 00 00 00 00 00 00 61 00 00 00 00 00 00 00 ........a.......
00007f09:ba96ec10 00 00 00 00 00 00 00 00 54 00 00 00 00 00 00 00 ........T.......
00007f09:ba96ec20 00 00 00 00 00 00 00 00 45 00 00 00 00 00 00 00 ........E.......
00007f09:ba96ec30 00 00 00 00 00 00 00 00 5f 00 00 00 00 00 00 00 ........_.......
00007f09:ba96ec40 00 00 00 00 00 00 00 00 32 00 00 00 00 00 00 00 ........2.......
00007f09:ba96ec50 00 00 00 00 00 00 00 00 30 00 00 00 00 00 00 00 ........0.......
00007f09:ba96ec60 00 00 00 00 00 00 00 00 31 00 00 00 00 00 00 00 ........1.......
00007f09:ba96ec70 00 00 00 00 00 00 00 00 34 00 00 00 00 00 00 00 ........4.......
00007f09:ba96ec80 00 00 00 00 00 00 00 00 5f 00 00 00 00 00 00 00 ........_.......
00007f09:ba96ec90 00 00 00 00 00 00 00 00 43 00 00 00 00 00 00 00 ........C.......
00007f09:ba96eca0 00 00 00 00 00 00 00 00 55 00 00 00 00 00 00 00 ........U.......
00007f09:ba96ecb0 00 00 00 00 00 00 00 00 5f 00 00 00 00 00 00 00 ........_.......
00007f09:ba96ecc0 00 00 00 00 00 00 00 00 31 00 00 00 00 00 00 00 ........1.......
00007f09:ba96ecd0 00 00 00 00 00 00 00 00 4e 00 00 00 00 00 00 00 ........N.......
00007f09:ba96ece0 00 00 00 00 00 00 00 00 5f 00 00 00 00 00 00 00 ........_.......
00007f09:ba96ecf0 00 00 00 00 00 00 00 00 4b 00 00 00 00 00 00 00 ........K.......
00007f09:ba96ed00 00 00 00 00 00 00 00 00 30 00 00 00 00 00 00 00 ........0.......
00007f09:ba96ed10 00 00 00 00 00 00 00 00 52 00 00 00 00 00 00 00 ........R.......
00007f09:ba96ed20 00 00 00 00 00 00 00 00 45 00 00 00 00 00 00 00 ........E.......
00007f09:ba96ed30 00 00 00 00 00 00 00 00 34 00 00 00 00 00 00 00 ........4.......
```

フラグが「H4PPY_C0DEGaTE_2014_CU_1N_K0RE4」