

Lab - Reverse Engineering crackme0x00 Using Ghidra

Overview

In our previous lab, we learned about some of the high-level features of Ghidra. We will continue where our previous lab left off by reverse engineering a simple executable, crackme0x00.exe.

A crackme is a small program designed to test a programmer's reverse engineering skills. Crackme files are created by other programmers as a legal way to crack software since no intellectual property is being infringed upon.

The crackme0x00.exe is a simple program that needs a password to open. Our file could be any executable that requires a password to launch.

Lab Requirements

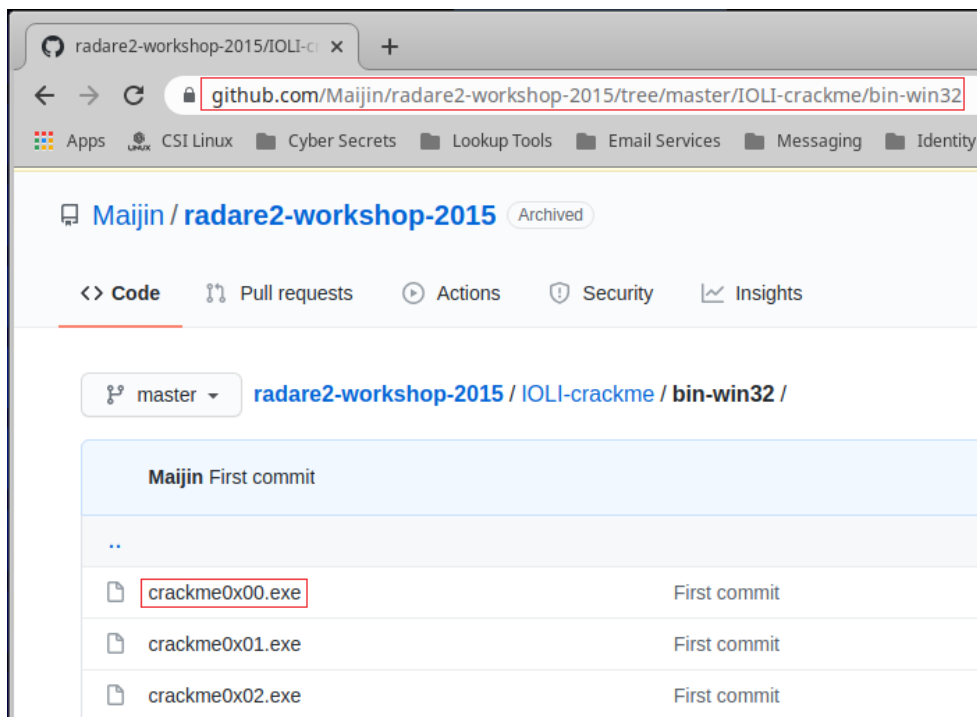
- One installation of VirtualBox with the extension pack.
- One virtual install of the latest version of CSI Linux.
- One virtual install of Windows 7
- VirtualBox network adapter set to NAT network.

Download crackme0x00.exe.

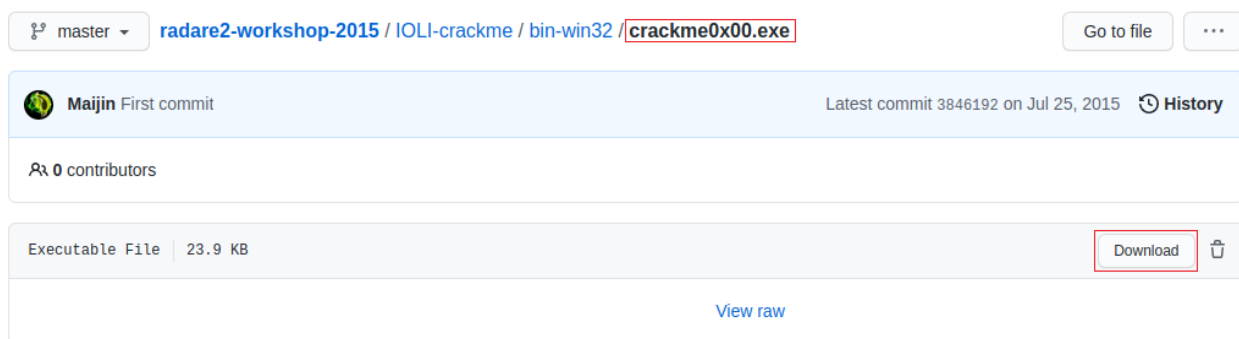
From your CSI Linux, open your default browser and copy and paste the following URL into the address bar.

<https://github.com/Maijin/radare2-workshop-2015/tree/master/IOLI-crackme/bin-win32>

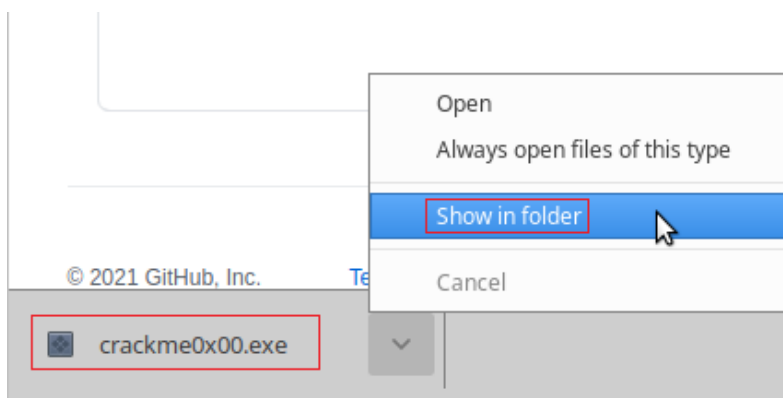
We will be analyzing the crackme0x00.exe file. From the list of crackme files, click on the file crackme0x00.exe.



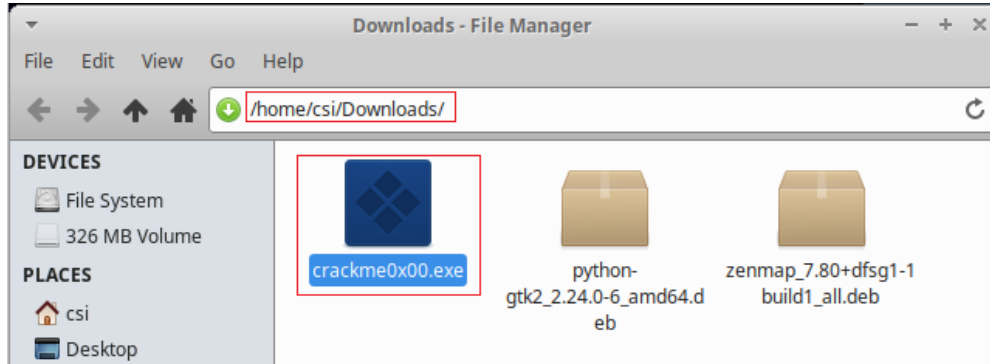
On the next page, click on the Download button.



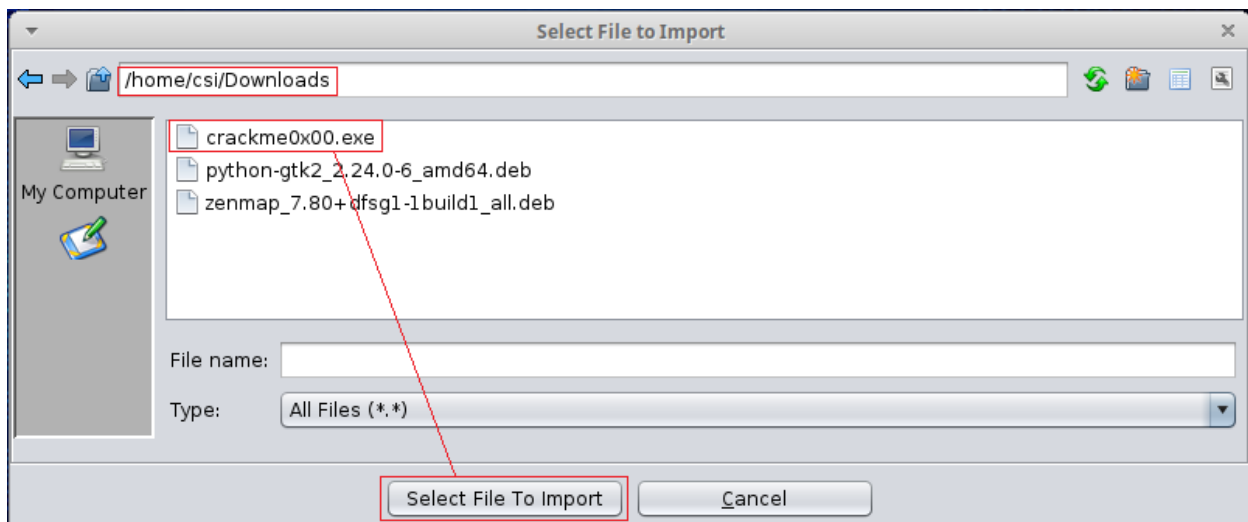
Once the file downloads, click the down arrow, and from the context menu, select **Show in folder**.



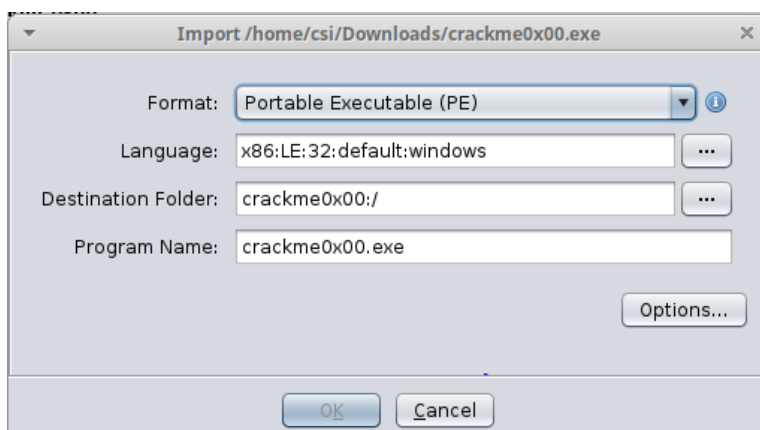
Take note that the file is in your Downloads Directory.



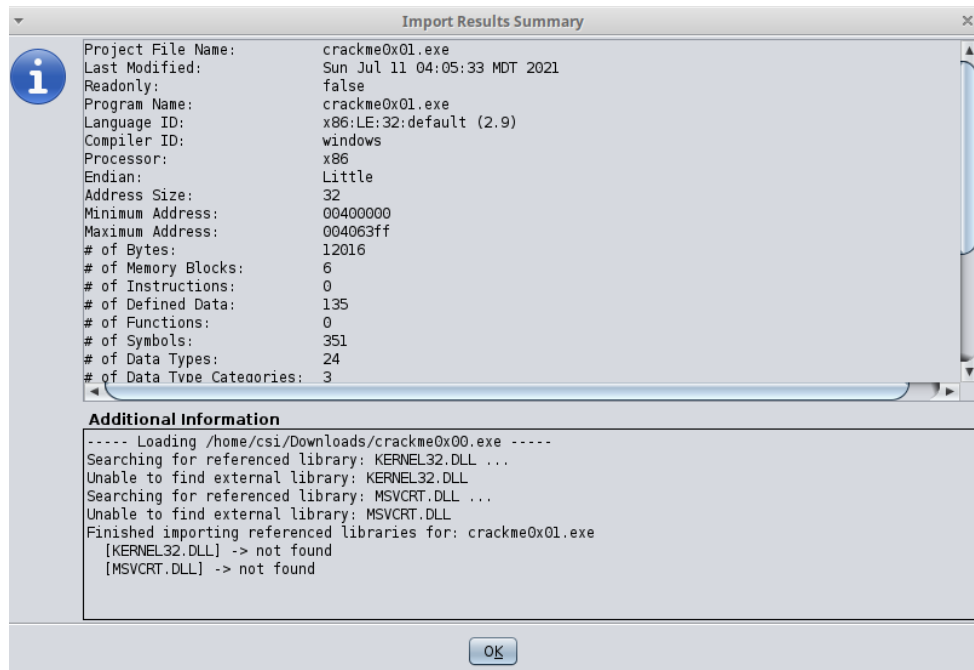
Close out your browser, the file manager, and return to Ghidra. There are two ways to import a file into Ghidra. You can click on File> Import File or open your Downloads directory and drag and drop the crackme0x00.exe into the project window.



When you begin the import process, Ghidra will respond with the following information. Click OK.

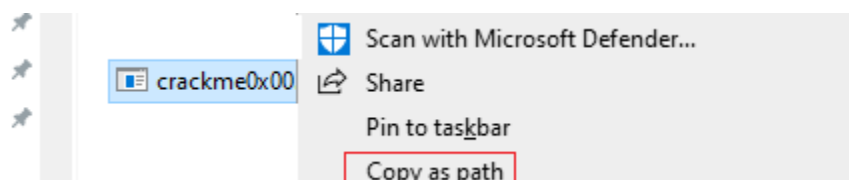


Once the file is imported, another screen loads with a summary of the import. Click oOK to close this screen.

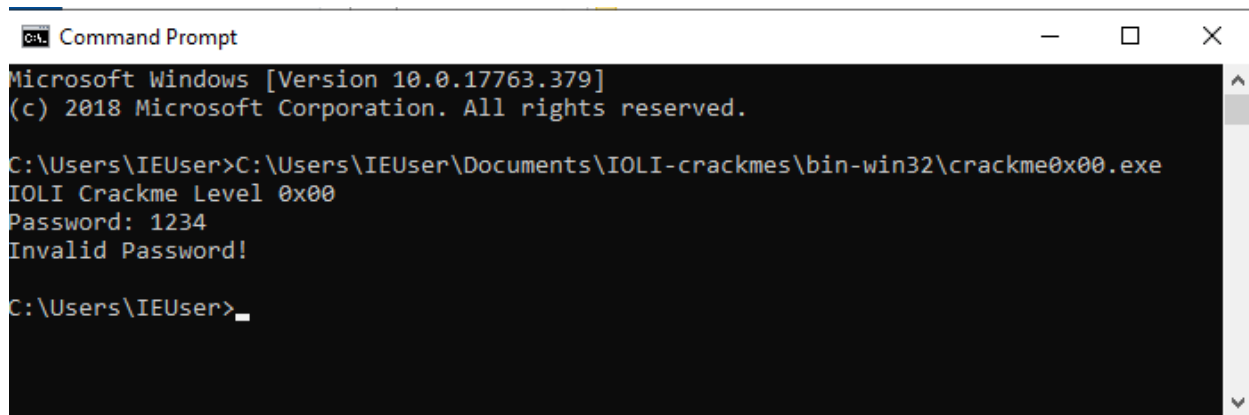


Examining crackme0x00.exe

You have the option of downloading **crackme0x00.exe** and running the program to examine its operation. In this example, I have downloaded the crackme0x00.exe onto a virtual install of either Windows 7. We can now run the program to see exactly what it does. Once the program has been saved to my virtual install of Windows 7, while holding down the shift key, right-click on the executable. This gives me an extended menu that includes 'copy as path.'



I next open a command prompt, and at the prompt, I right-click pasting in the path of the executable. When I press enter, and I am prompted for a password. I try and guess the password, but it fails. So our job is to reverse engineer this program to find the correct password.



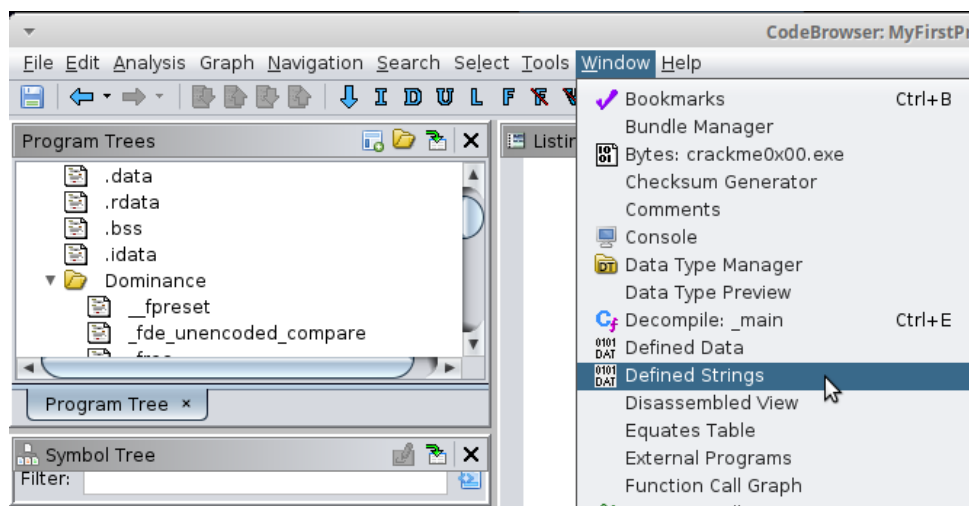
```
Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\IEUser>C:\Users\IEUser\Documents\IOLI-crackmes\bin-win32\crackme0x00.exe
IOLI Crackme Level 0x00
Password: 1234
Invalid Password!

C:\Users\IEUser>
```

Note that part of the assembly code has a defined string of ‘Password:’ in it. Thus, we can search the program for a “Defined String” of ‘Password:’

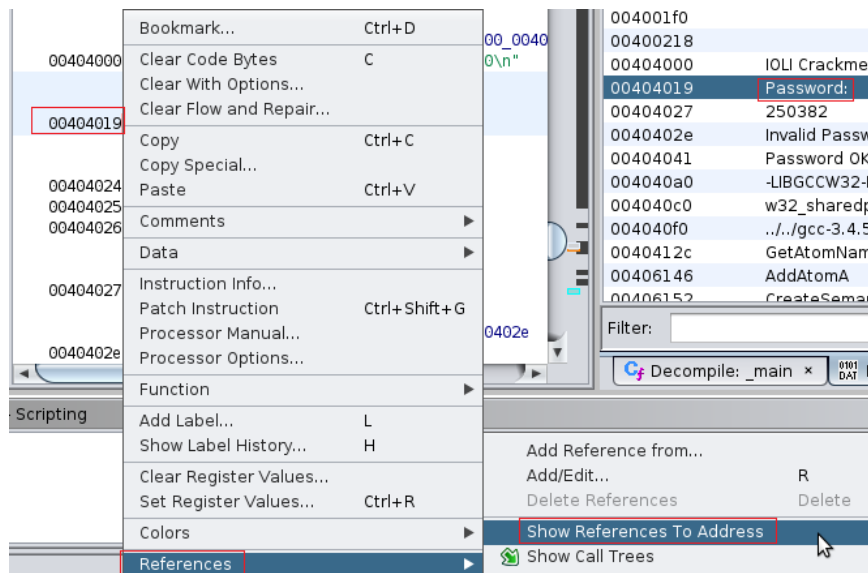
From the Taskbar in Ghidra, click the Window tab, and From the context menu, select “Defined Strings.”



A new window opens, showing you all the Defined Strings present in the assembly. Find and click the predefined string, ‘Password:’

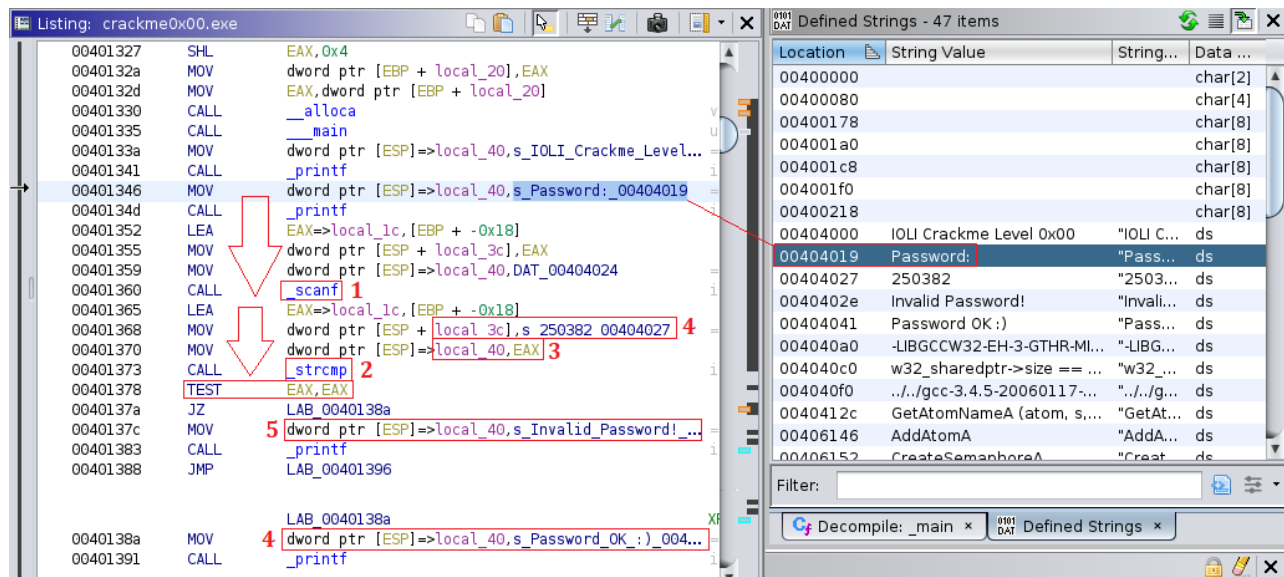
To the right, you see the Listing Window. This shows where the string sits in the assembly code.

In the listing pane, right-click on the exact address (00404019) assigned the defined string, Password: From the context menu, select “References > Show References to Address.”

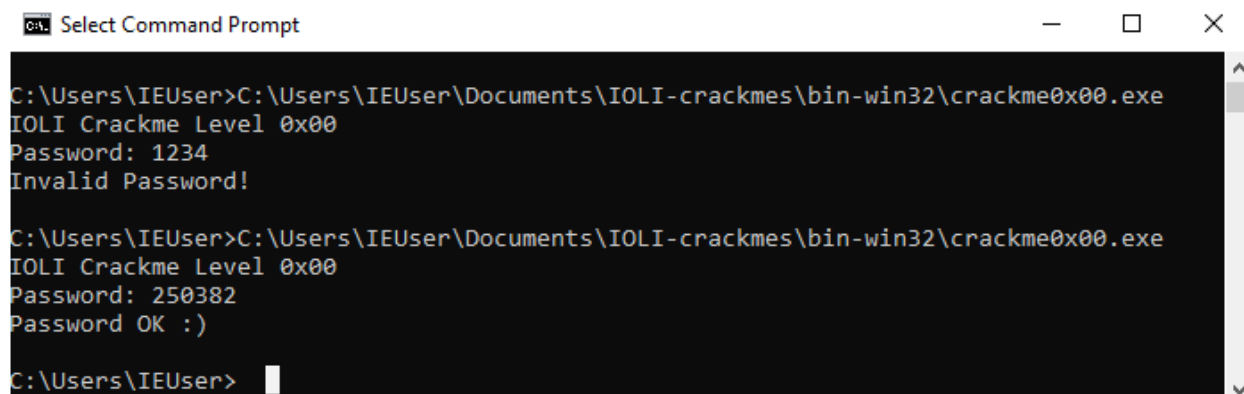


The references to address are shown in the Listing Window.

Notice that after the reference to “Password:” there is a call to `scanf` (1) to receive the user input and a call to `strcmp` (2) after that. We see that the user input gets stored in EAX and placed into a local variable called `local_40` (3). The string “250382” is also stored into a local variable called `local_3c`, (4) from here; both are passed to `strcmp` for comparison (2). The result of this comparison is checked against the value zero, and if it is equal to zero, then the text “Password OK :)” (4) is printed. Otherwise, it takes the jump and prints the text “Invalid Password (5).



Back at the crackme0x00.exe password prompt, we type in 250382, and we are in.



```
C:\Users\IEUser>C:\Users\IEUser\Documents\IOLI-crackmes\bin-win32\crackme0x00.exe
IOLI Crackme Level 0x00
Password: 1234
Invalid Password!

C:\Users\IEUser>C:\Users\IEUser\Documents\IOLI-crackmes\bin-win32\crackme0x00.exe
IOLI Crackme Level 0x00
Password: 250382
Password OK :)

C:\Users\IEUser>
```

Summary –

In this lab, we reversed engineered a simple executable, **crackme0x00.exe**, by searching for a defined string discovered when we launched the program. Going forward will continue to work with other crackme files to improve upon our reverse engineering skills.