Reversing & Exploiting with free tools

This is the beginning of a new training which will be focused in reversing and exploitation for Windows using free and easy to get tools (IDA FREE, Radare, Windbg, X64dbg, Ghidra, etc).

This time no paying tool will be used, and the idea is not to generate any kind of competition between the different free tools, we will learn how to use each one, and in my case learn how to use Radare and Ghidra, both are tools I don’t use in my every day and I’ll try to use them as best as I can, as I’m not an expert in any of those two tools, if anyone sees any error or better way to use them, it would be nice to let me know, and I will correct it and I will learn too.

This training will start with the tool installation, later a little bit of vulnerability theory, and finally very easy examples of exploitation that will raise in complexity with the course. Introduction to assembly and assembly instructions will not be covered because it was covered in the “REVERSING CON IDA PRO” training, previous trainings and many other places on Internet, here we will straight to reversing and exploitation. Please refer to all previous trainings if you have doubts with the assembly instructions.

To set up my work environment, first thing I’ll do is to install the tools.

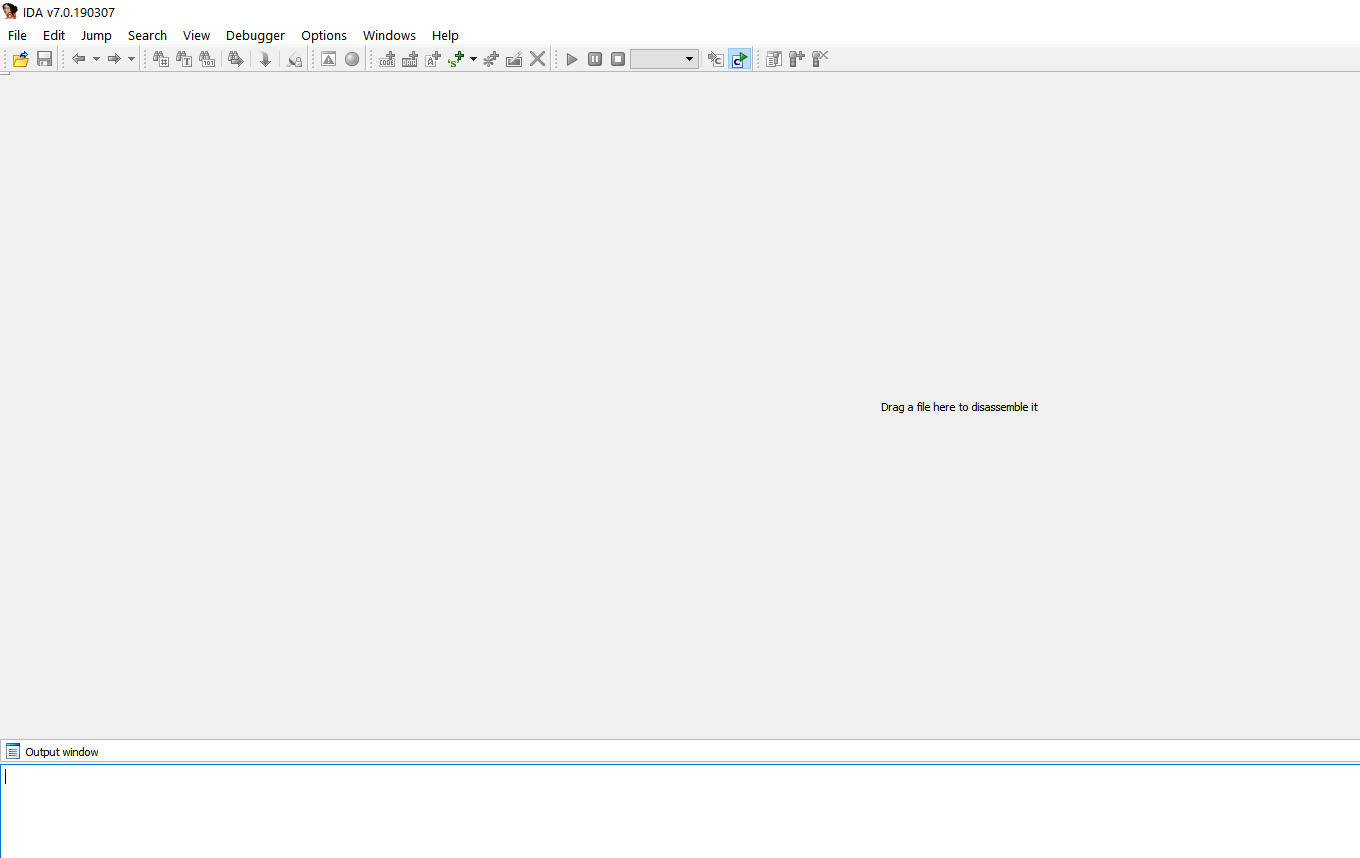
**INSTALLING IDA FREE**

IDA FREE can be downloaded from:

<https://www.hex-rays.com/products/ida/support/download_freeware.shtml>



From there, download the file idafree70\_windows, it’s pretty easy to install, just follow the installer instructions and you’ll get IDA FREE running in your machine quickly.



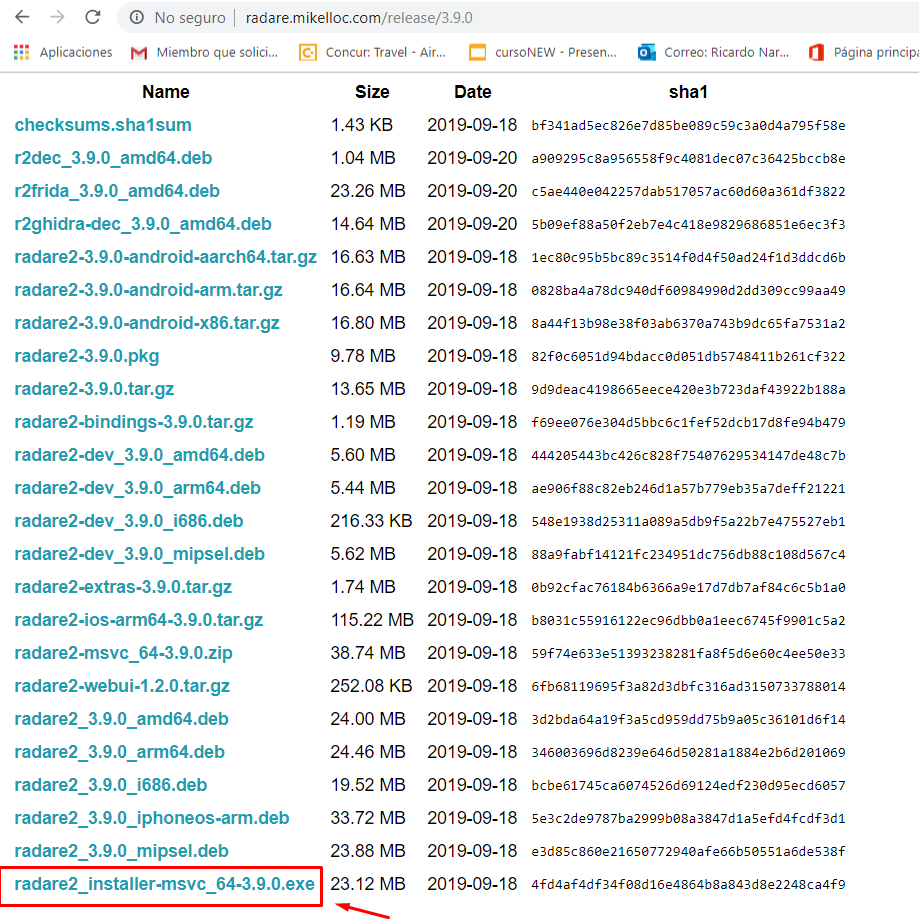
**INSTALLING RADARE**

Next step will be to install radare2. Radare can be downloaded from: <https://github.com/radareorg/radare2/releases>

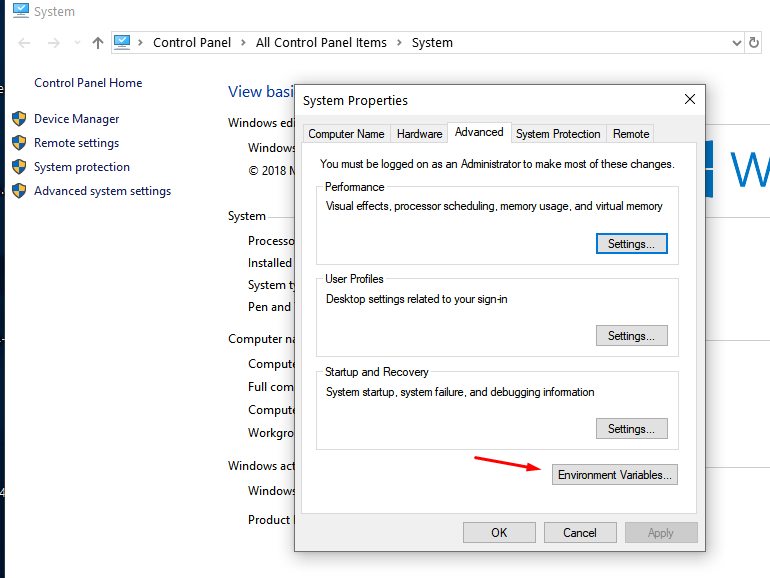
There, we go to the last release.



At the time of writing this document last release is the 3.9.0, in future there will be one newer, it doesn’t matter just download the download the newer installer for Windows.



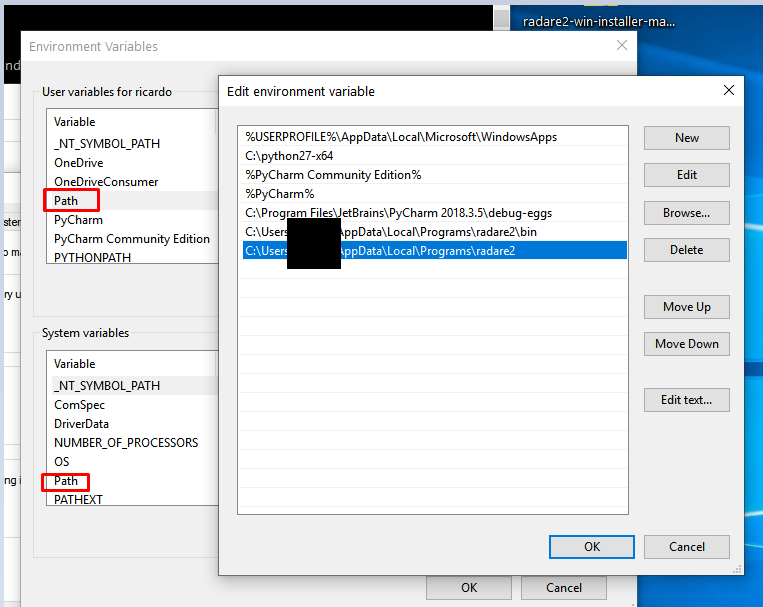
Installing is also simple, once installation has finished just include the path, where radare was installed, into the environment variables.



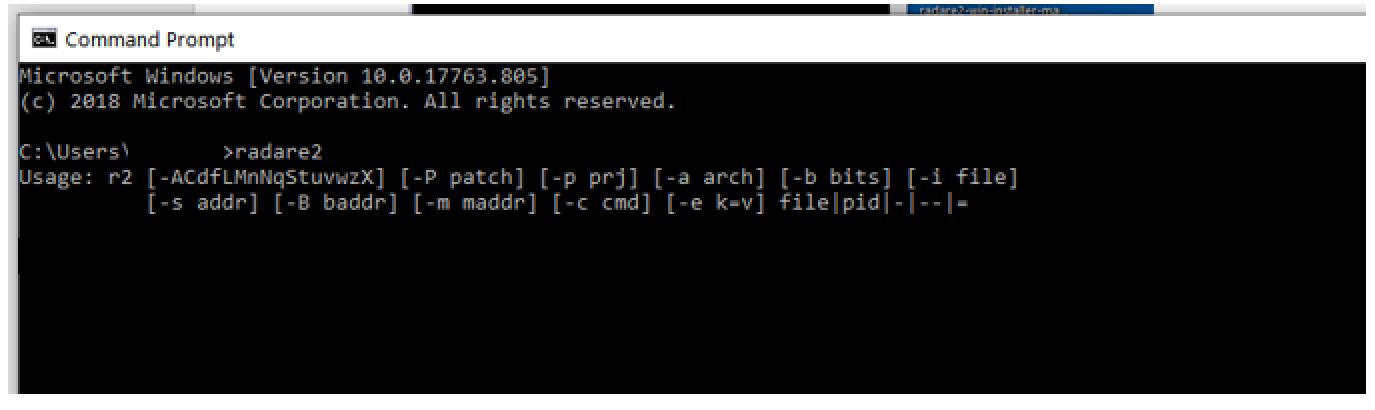
Inside of the environment variables, we go to the variable path and include these two lines (write your own paths if you installed elsewhere):

C:\Users\<user\_name>\AppData\Local\Programs\radare2

C:\Users\<user\_name>\AppData\Local\Programs\radare2\bin

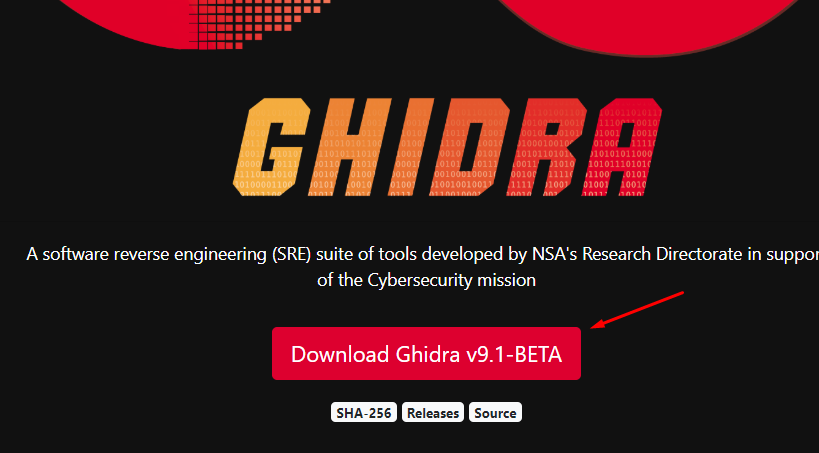


The idea is that from a command prompt (CMD) we can invoke the command radare2, and windows should recognize it.



**INSTALLING GHIDRA**

Installing GHIDRA is pretty easy too, just go to their webpage: <https://ghidra-sre.org/>

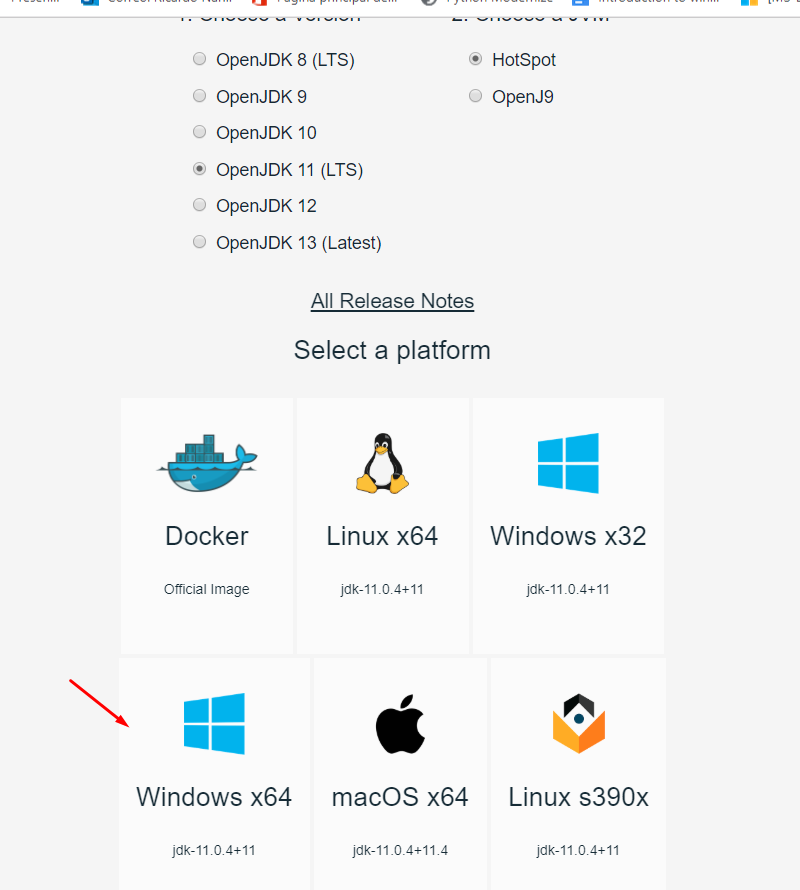


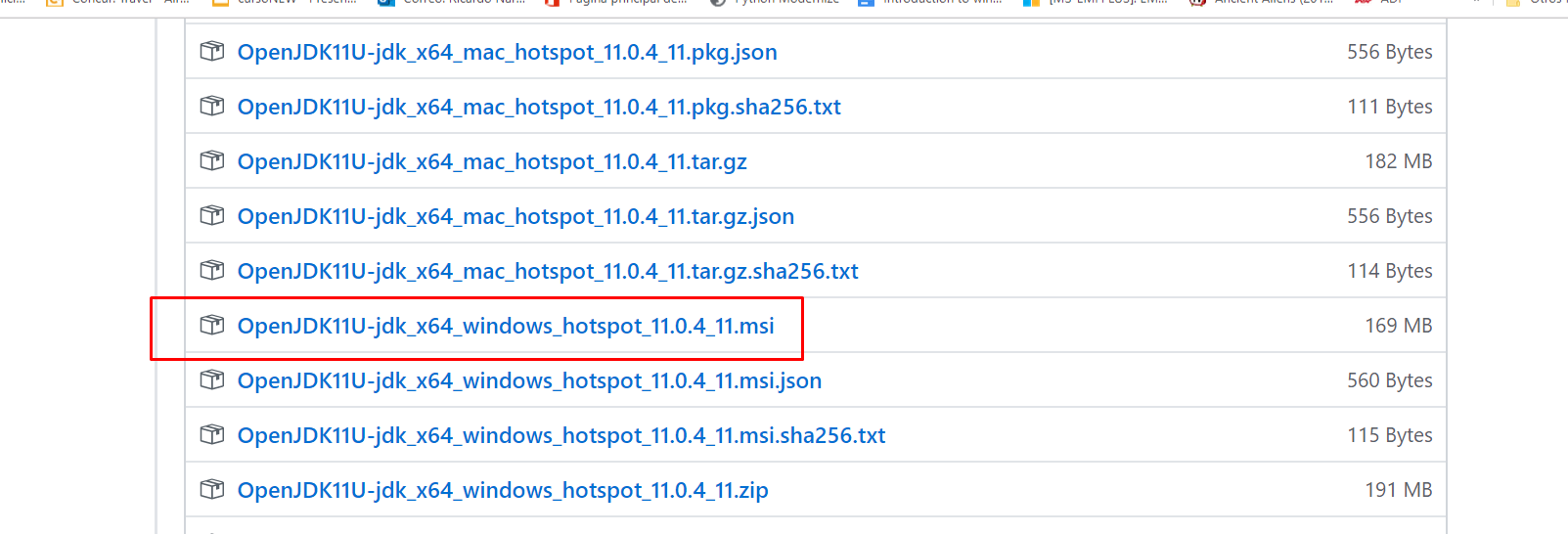
There you just have to download the zip (in my case at time of writing this paper it was ghidra\_9.1-BETA\_DEV\_ 20190923.zip) and decompress it wherever you want, I created a Virtual Machine in VMWARE, without internet connection, I just use it for GHIDRA, it’s not that I distrust the NSA, but it’s better to work in that environment.

Once the tool is decompressed, we have to install a compatible Java, in the Oracle webpage you can download it, or it’s possible to download it from other places (<https://www.filehorse.com/es/descargar-java-development-kit-64/36775/>) in my case I downloaded this installer, once the installer has finished, include java path where is located the java executable (usually the bin path) to the path environment variable as seen before.

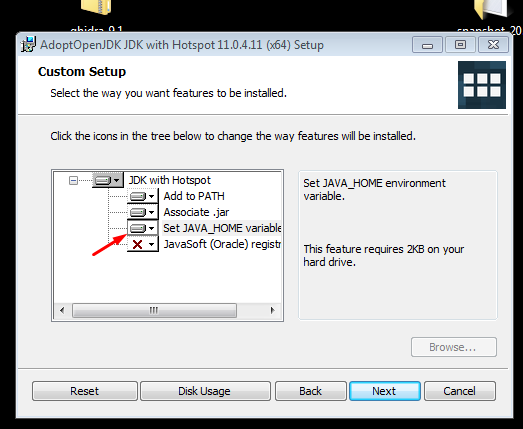
Version 11 is the recommended for compatibility by the GHIDRA web page.

Other people who used GHIDRA recommend the version 11 of the OpenJDK , you can download it from here: <https://adoptopenjdk.net/releases.html?variant=openjdk11&jvmVariant=hotspot>

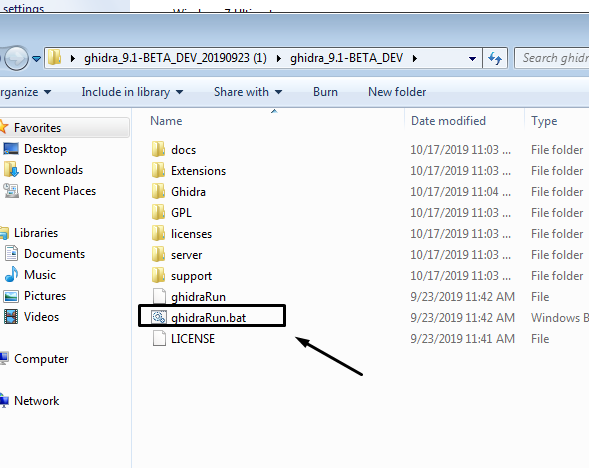




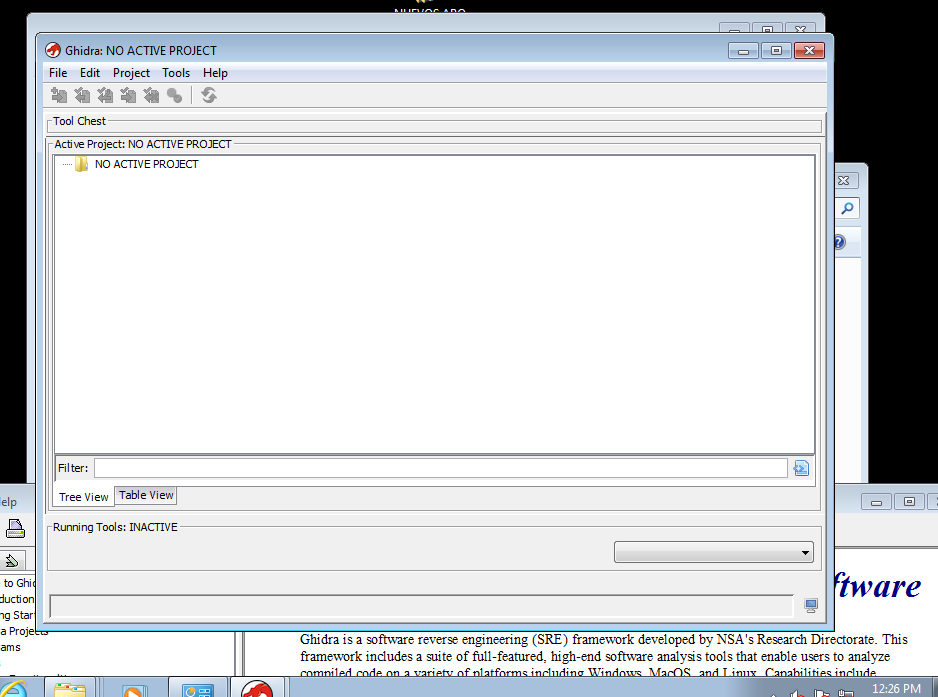
While installing with the OpenJDK installer, it’s possible to automatically add it to the variable PATH:



Once Java is installed in your environment, we can run GHIDRA:



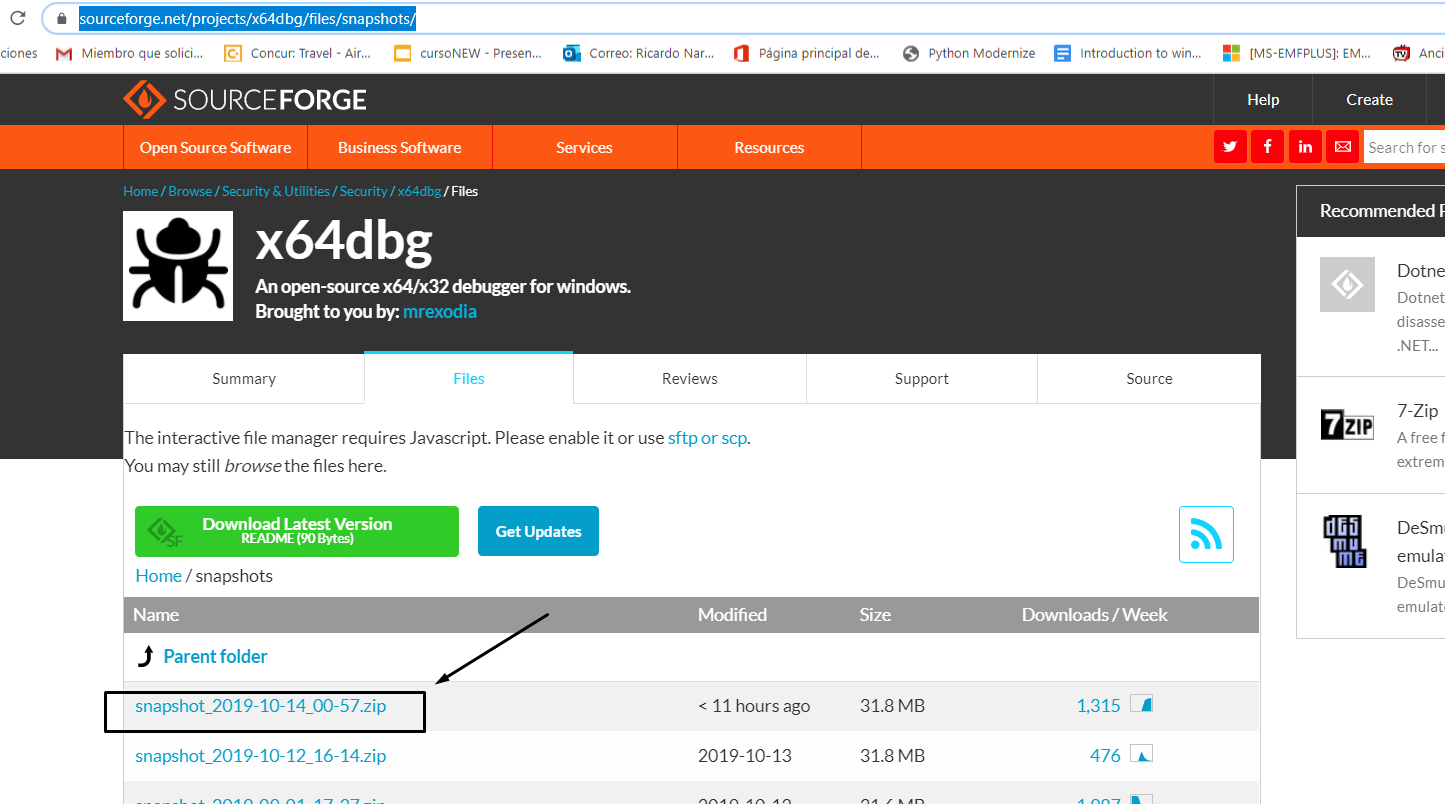
It’s just clicking bat file, and GHIDRA boot up:



Another tool installed!

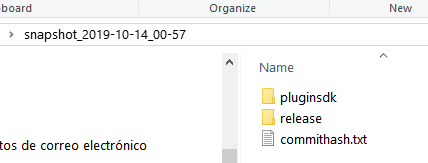
**INSTALLING X64DBG**

Regarding this program, almost everyday we have new snapshots, just go to their sourceforge web page and install the last one: <https://sourceforge.net/projects/x64dbg/files/snapshots/>

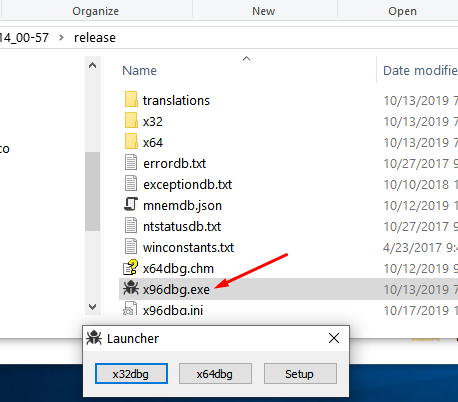


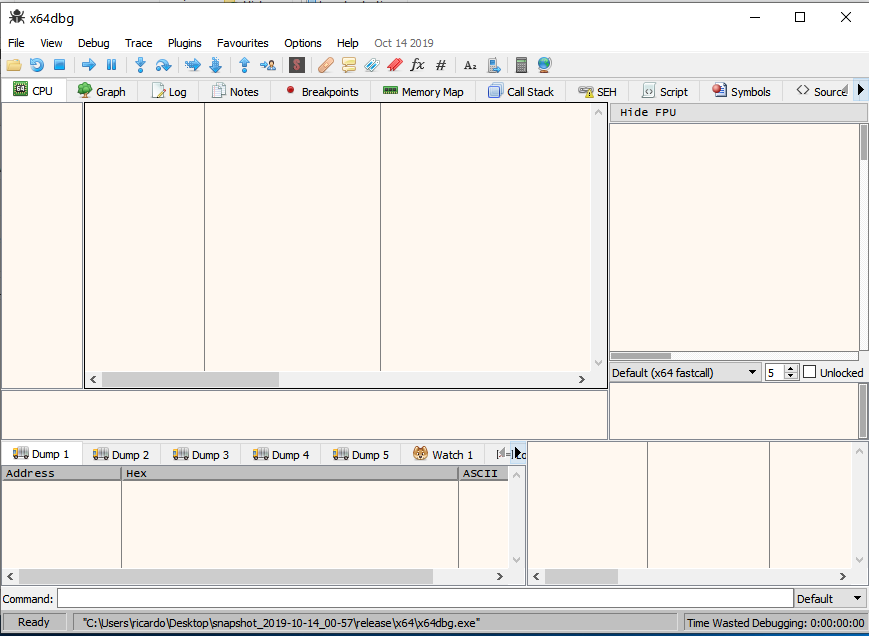
In my case it was snapshot\_2019-10-14\_00-57.zip.

Once we’ve unzipped the file, move to the release folder:



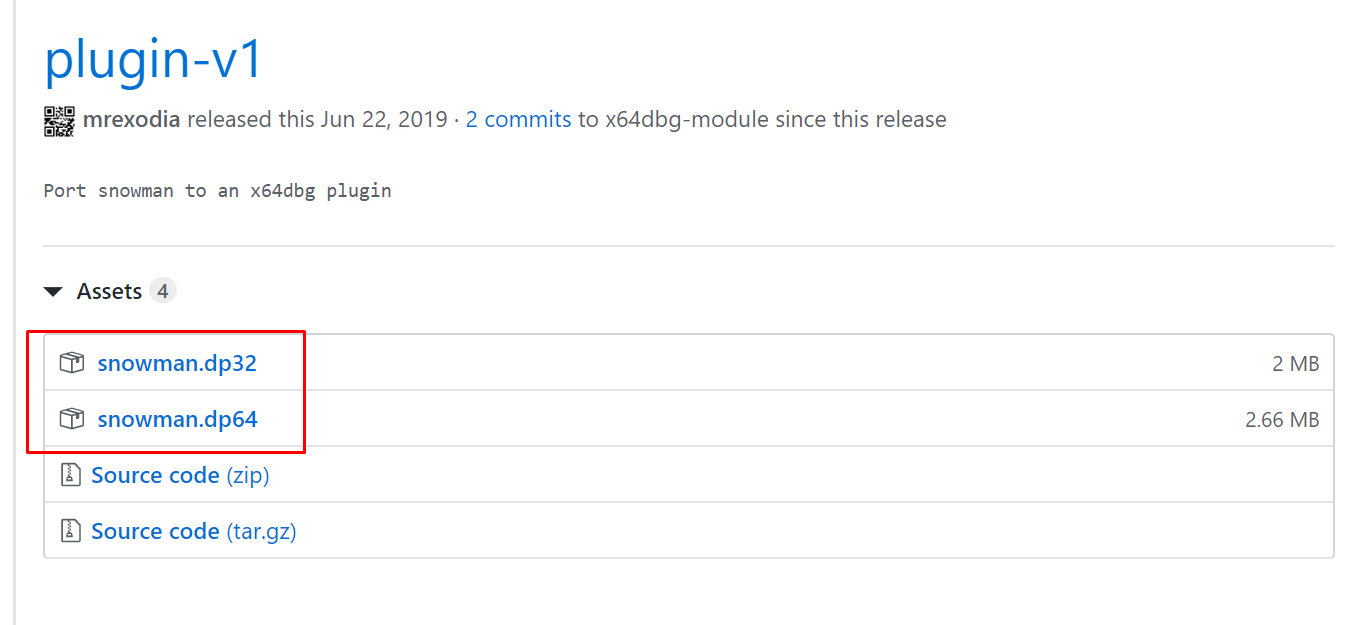
When you run it with administrator privileges, a launcher appears for you to choose which version you would like to run (32 or 64 bits).



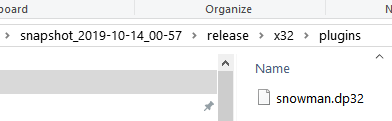


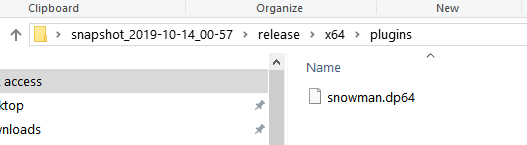
Snowman was part of x64dbg, but now it’s a plugin we can download and install, this plugin will decompile our binaries in x64dbg as IDA FREE does not come the Hex-rays decompiler, just download it, and copy it inside of plugins folder.

<https://github.com/x64dbg/snowman/releases/tag/plugin-v1>



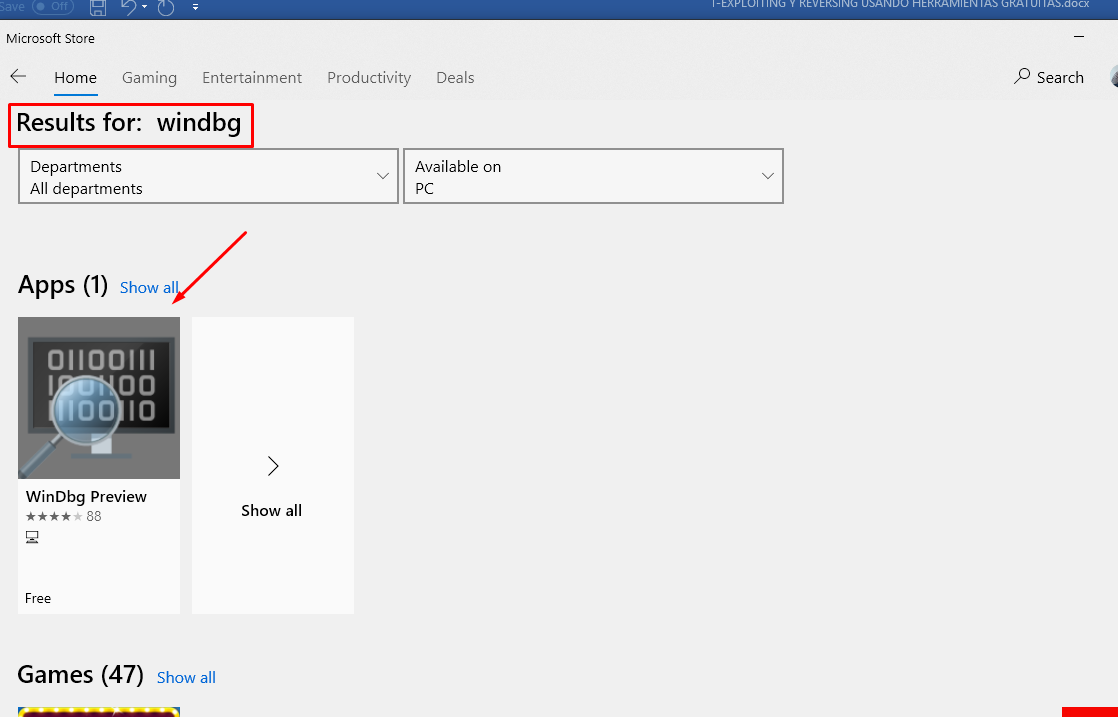
The one of 32 bits goes to 32 bits plugins folder, and 64 bits goes to the 64 bits version plugins folder.





**INSTALLING WINDBG**

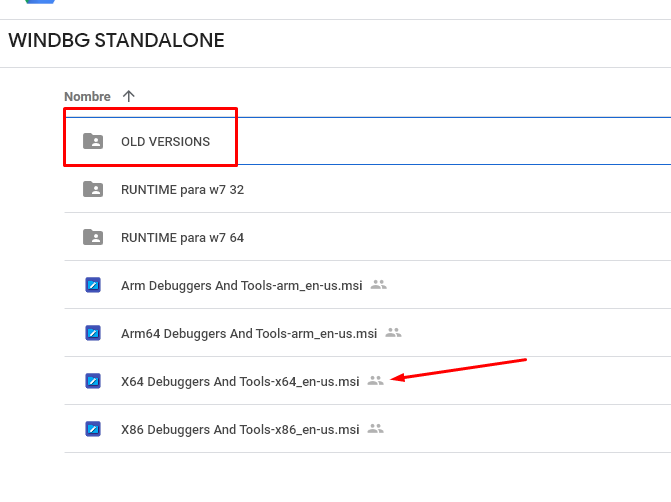
If you have Windows 10, to install windbg you just have to go to the Microsoft STORE and search for WINDBG, once it appears, we can install it from there.



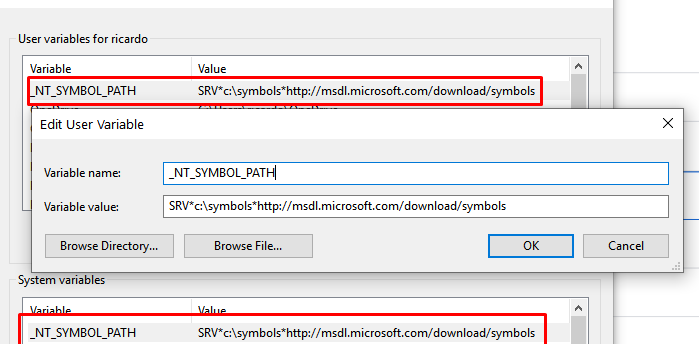
From there WINDBG PREVIEW is installed automatically, this is the newest version. If you have the Windows 7, you’ll have to install a previous version of Windbg.

[*https://drive.google.com/open?id=1UEPBecOoir-nhyRK-RHOAhRq05ZtlW1y*](https://drive.google.com/open?id=1UEPBecOoir-nhyRK-RHOAhRq05ZtlW1y)

There you have some older versions:



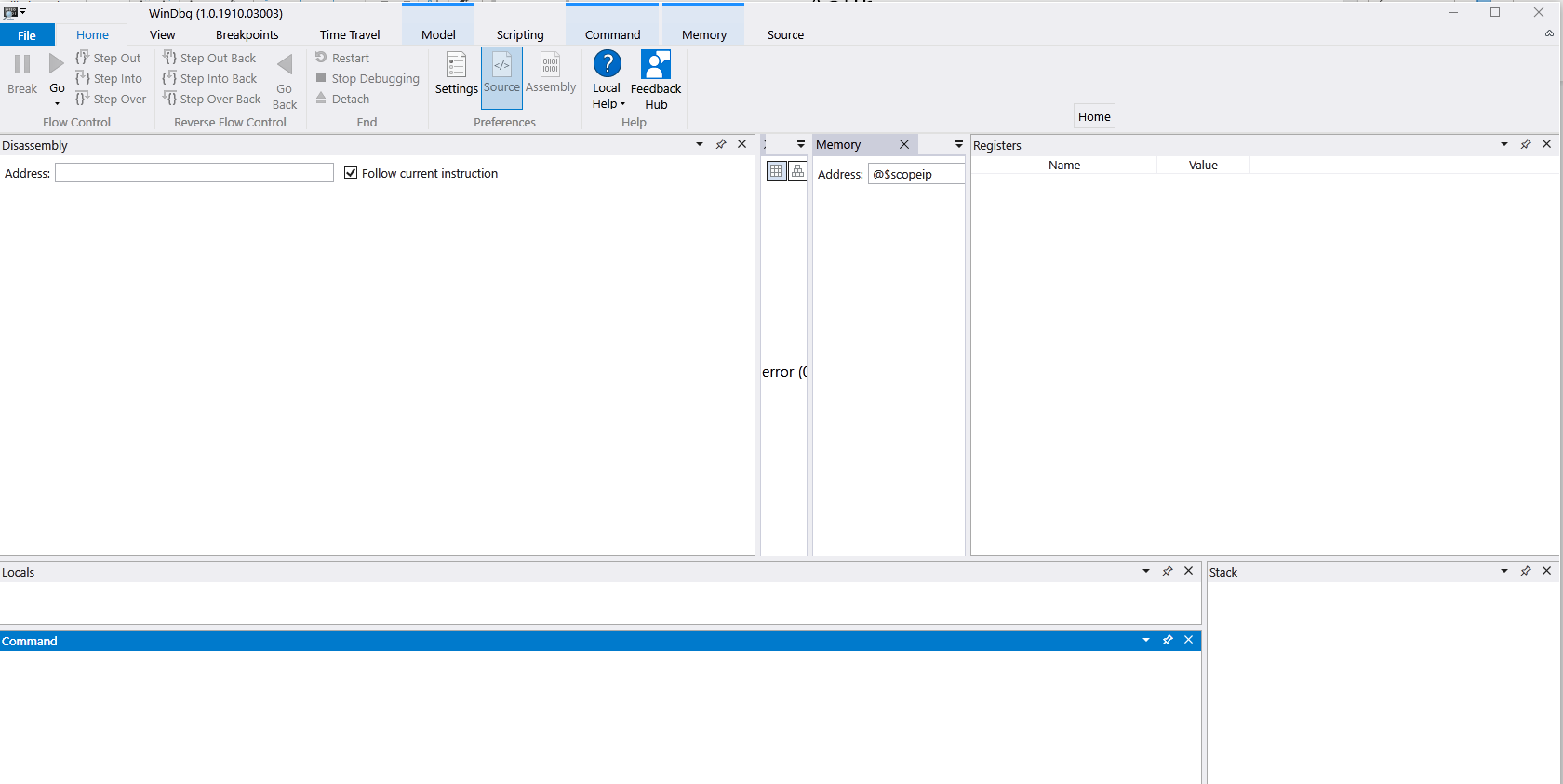
Next step, configure symbols for WINDBG, create the folder symbols in “C:\” and then go to environment variables and create the variable \_NT\_SYMBOL\_PATH.



As value write:

***SRV\*c:\symbols\*http://msdl.microsoft.com/download/symbols***

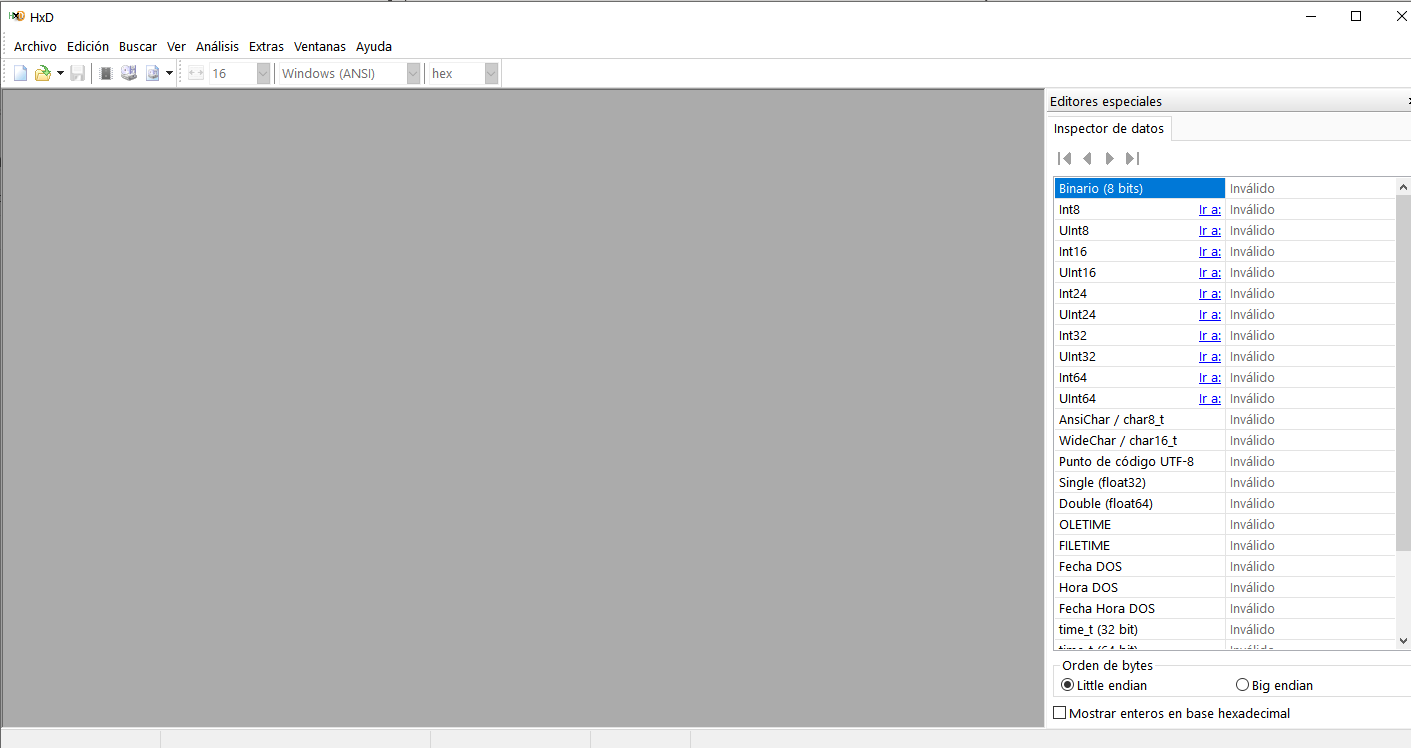
With this we already have installed WINDBG PREVIEW.



**INSTALLING A HEXADECIMAL EDITOR**

For this step, we can just download HxD that is free and it works well.

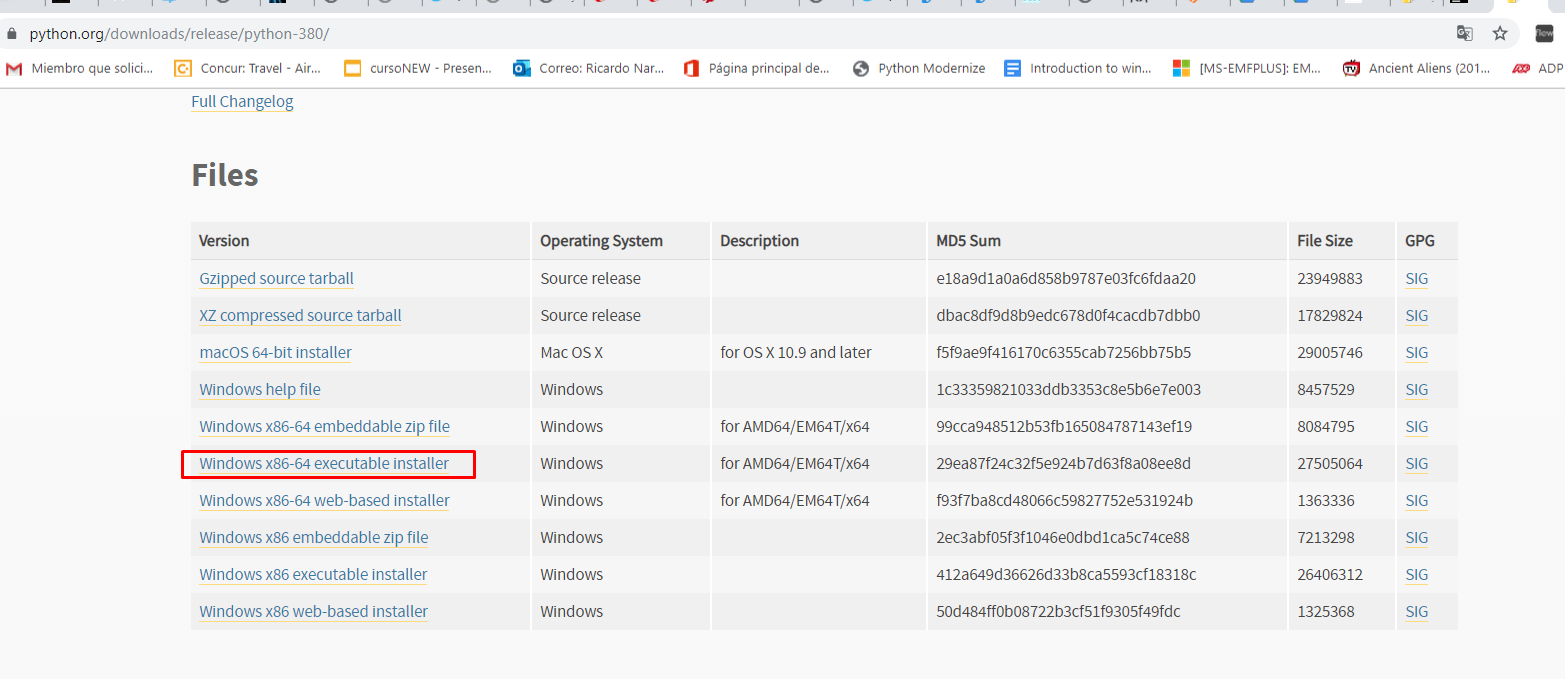
<https://mh-nexus.de/en/hxd/>



**INSTALLING A PYTHON**

We will install Python 3 that is the newest version, this will be helpful to create the exploits for each exercise.

<https://www.python.org/downloads/>



Just download the last version. In the time of writing this document it was:

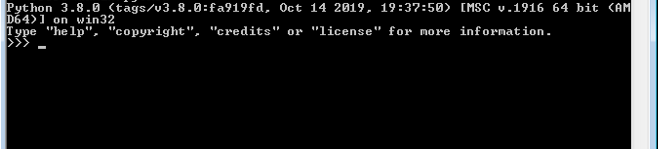
Python-3.8.0-amd64.exe

To aggregate Python.exe to the PATH environment variable there’s an option in the installer, just click it and it will be done automatically.

Usually the installation path will be:

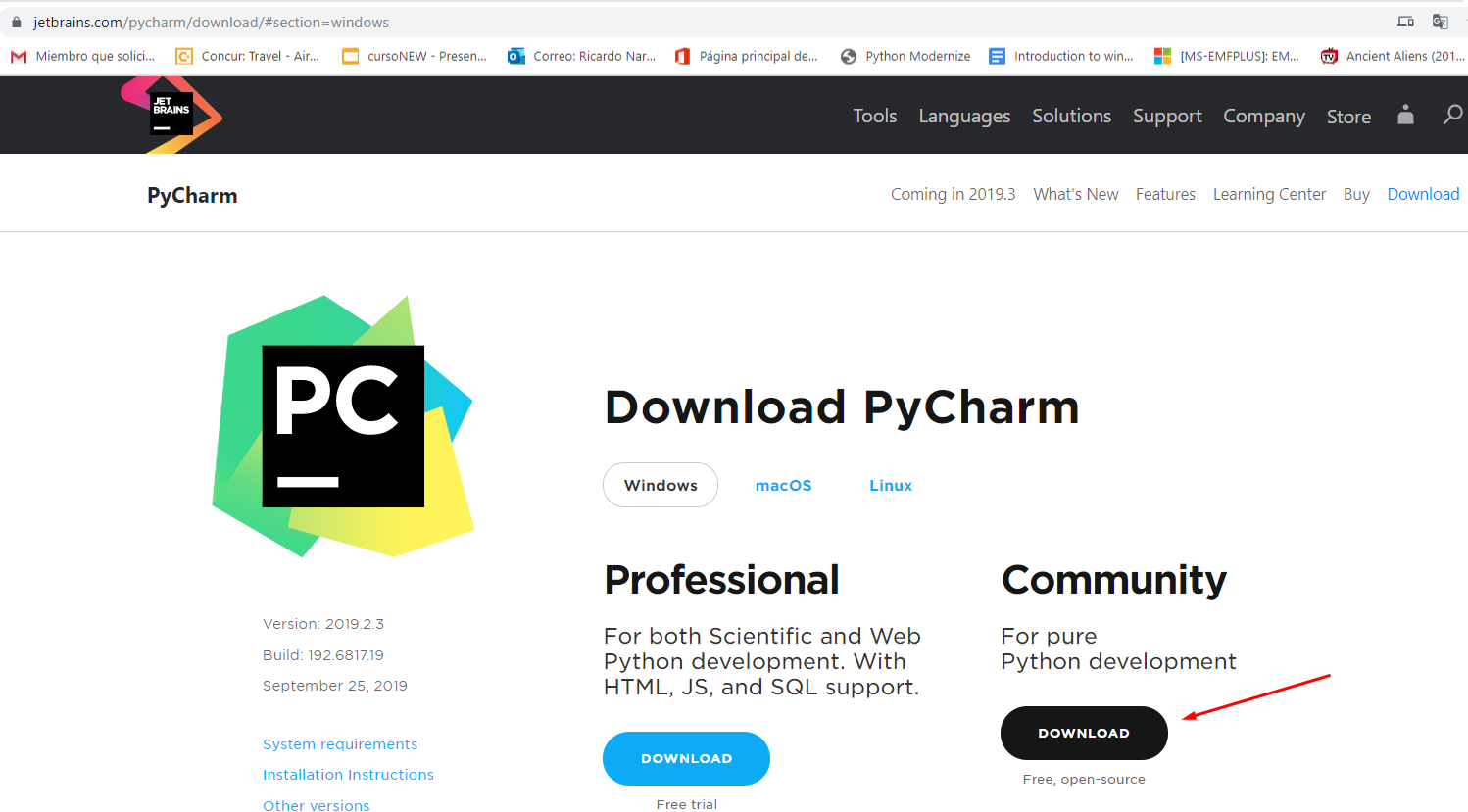
C:\Users\**XXXXX**\AppData\Local\Programs\Python\Python38

(Just realize that Python38 could be different in your case). The nuts and bolts of this is that we should be able to execute Python at the Command prompt.

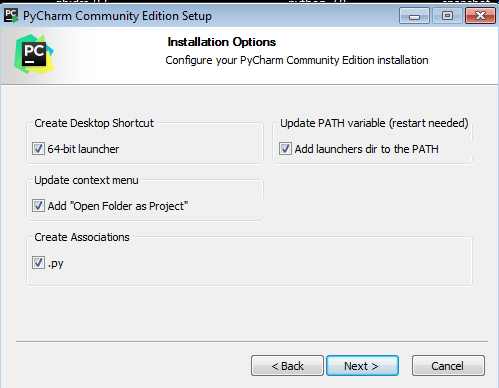


**INSTALLING A PYCHARM COMMUNITY**

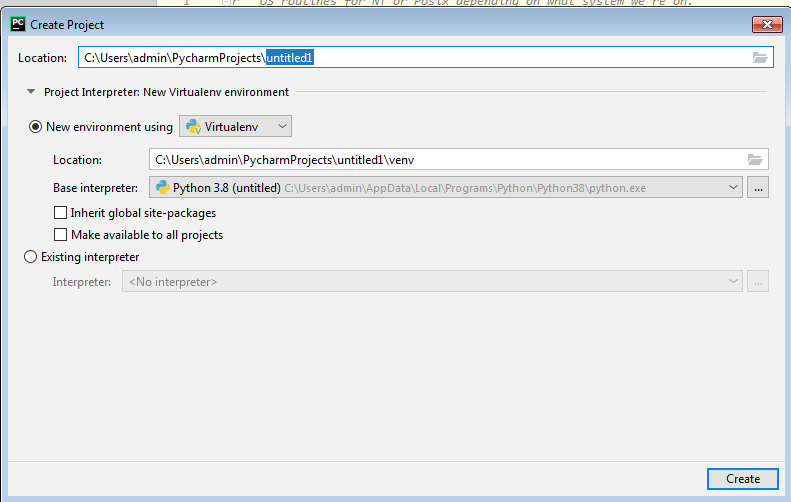
This will be our Integrated Development Environment (IDE) for python. Just go to jetbrains web page: <https://www.jetbrains.com/pycharm/download/>



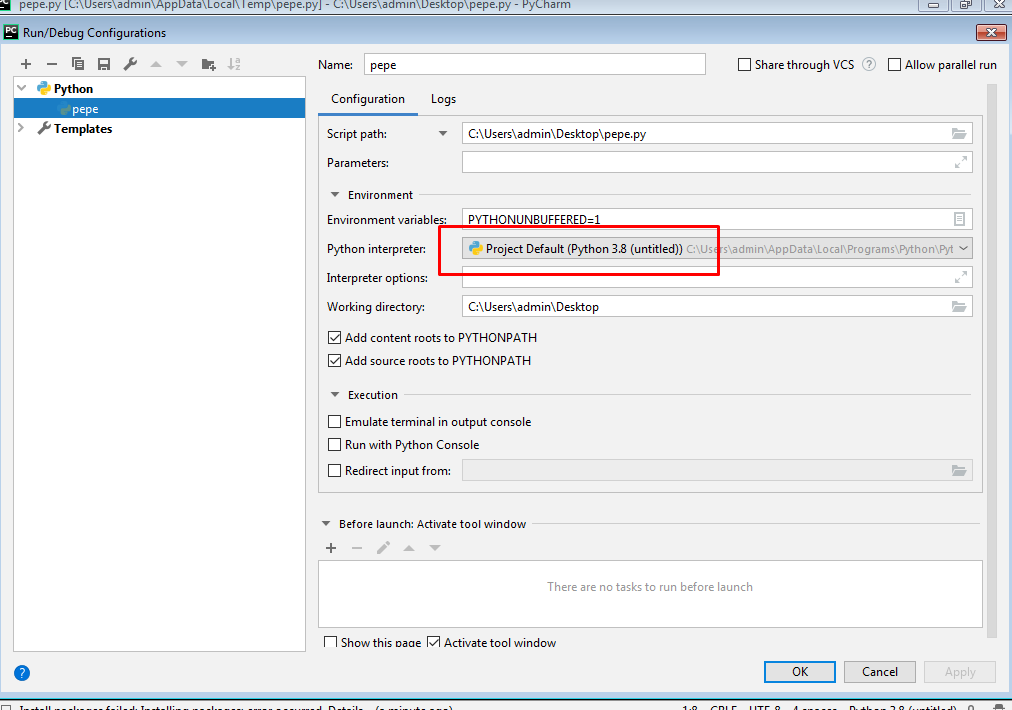
In my case the last pycharm version is es pycharm-community-2019.2.3.exe, for you it can be different one:



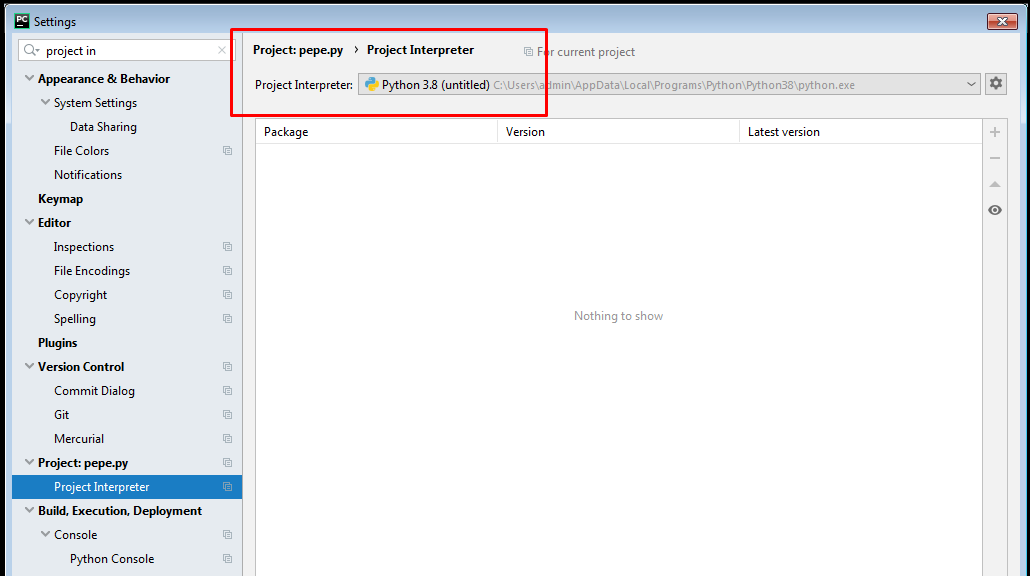
Set all these options, so it will be included in PATH environment variable. Once installed, create a new project:



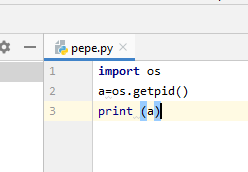
In RUN->DEBUG configuration check that “Python interpreter” option points to our Python interpreter (In my case Python 3.8).



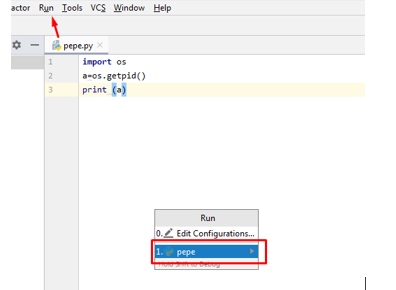
Also, in SETTINGS search for PROJECT INTERPRETER and check that Python 3.8 is detected (or the version you decided to install).



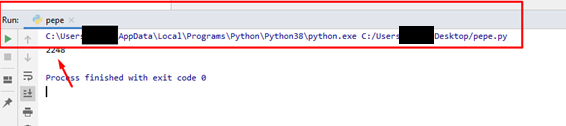
This will allow you to create .txt files, change to .py extension and move them into Pycharm.



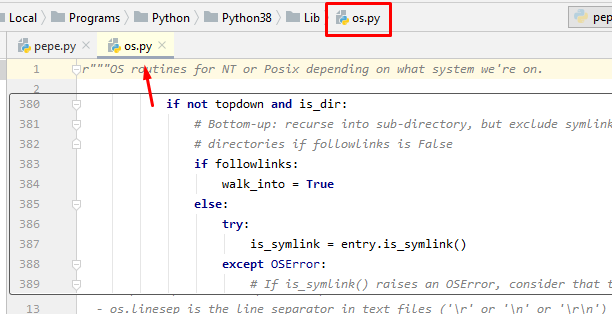
For example, a sample file named pepe.txt has been renamed into pepe.py and dragged and dropped into Pycharm. When you click “Run,” the next screen should appear:



And finally, the the console of pycharm screen should print the next:



Pycharm once we are typing should autocomplete, and if we point with the mouse the word “os”, if we press CTRL and click, pycharm should take us to the code of “os” python library.



With this we finish the part 1, we’ve installed the tools for our exploiting environment. In part 2 we will start with a little bit of theory about buffer overflow and with first simple exercises, some of them will be explained step by step and the others will be left as exercises for you to resolve them (I encourage you to try them, is the best way to learn).

See you in part 2 of the training

Ricardo Narvaja ([www.ricardonarvaja.info](http://www.ricardonarvaja.info/))

Translated to English by Fare9.

18/10/2019