

REVERSING WITH IDA PRO FROM SCRATCH

PART 22

DIFFERS

A differ is a program that from two consecutive versions of the same program, tries to match the functions despite the changes and tries to show which functions were changed and where.

Obviously this work is not simple, especially when from one version to another there have been many changes which go from the application of some security patch to solve some vulnerability, as well as there may be improvements in the program, added in the interface, general changes , etc.

Since we have to work with differs we know that the bigger and more changes in the executable, the more ungrateful is the work since the Differ makes some mistakes when matching.

Let's see the three best known differs we will use in general, to install and know them, each has its strength and weakness. Sometimes more than one has to be used in complex cases to try to clear up.

The first one we are going to install is the BINDIFF.

<https://www.zynamics.com/software.html>

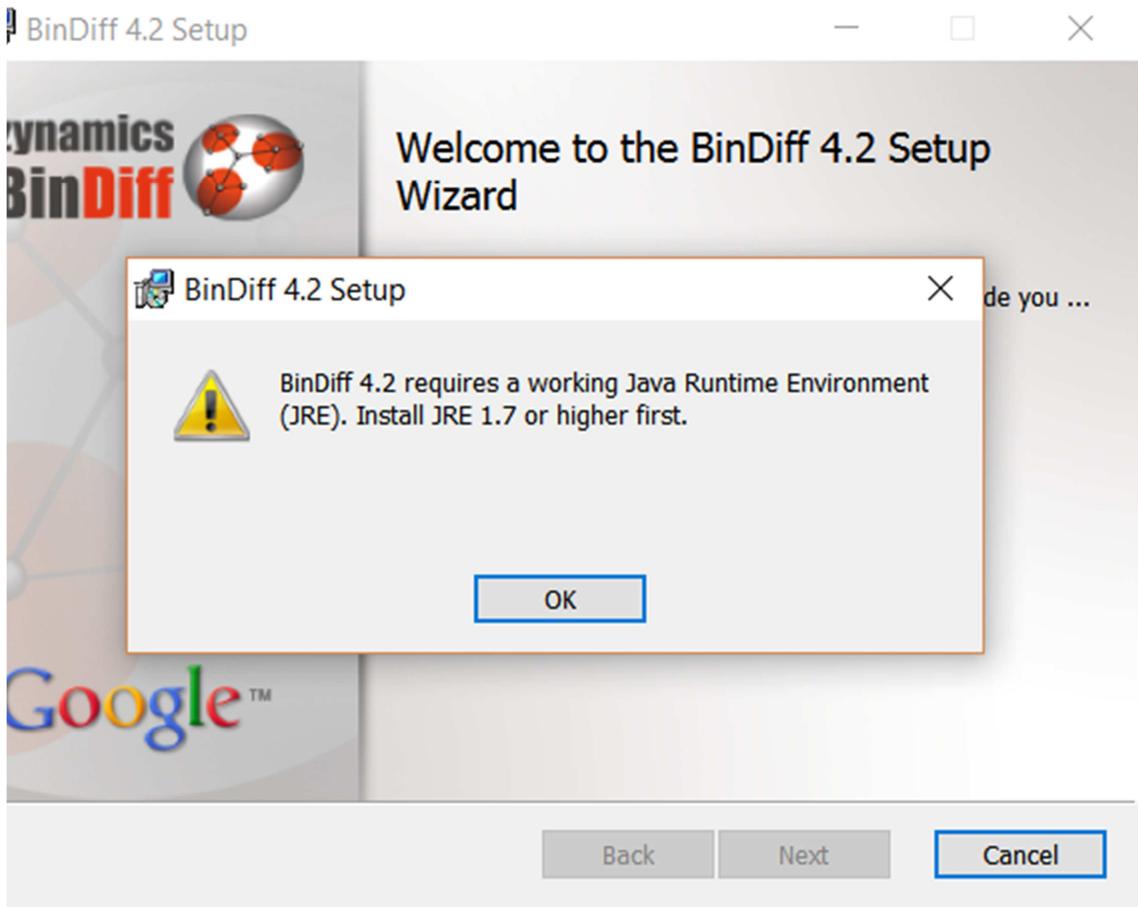
zynamics BinDiff | Download now for free.

By clicking this box, you are indicating that you have read and consent to be bound by the terms of the [End User License Agreement](#). Such consent is a prerequisite to downloading the BinDiff software. If you have not read the terms or do not agree to these terms, then do not click the box and do not download the software.

| Filename | Size | SHA1 |
|--------------------------------|------|---|
| bindiff420-debian8-amd64.deb | 15M | 38fbea8070495fc8730d7c86eae03bc68fde291f |
| bindiff420-debian8-i386.deb | 15M | 49cdd6ae7ebef5b1813a5fcfaaae9fde19005c824 |
| bindiff420-win-pluginsonly.zip | 5.8M | e2b786d405aac23aced989e02080dd69c18ab75e |
| bindiff420-win-x86.msi | 22M | 89f2eadc6582d4acc1e78db3617b5fba3eced0f |
| bindiff-license-key.zip | 990 | 95715a8bd7469106fc60b03f94f3cc87604e354c |

Note: We do not offer support. If you do contact us with bugs, feature requests or general questions, we will decide on a case by case basis on how to respond.

From there it can be downloaded accepting the conditions.



Well, download java if we do not have it.

The screenshot shows the Oracle Java SE Downloads page at www.oracle.com/technetwork/java/javase/downloads/index.html. The page features a sidebar with links like "tagazine", "FORUMS", "Java Mag", "Java.net", "Developer", "Tutorials", and "Java.com". The main content area includes a notice about Java SE 8u111 security fixes and a yellow box highlighting an "Important planned change for MD5-signed JARs". Below these are links for "Installation Instructions", "Release Notes", "Oracle License", "Java SE Products", "Third Party Licenses", "Certified System Configurations", "Readme Files" (with sub-links for "JDK ReadMe" and "JRE ReadMe"), and download buttons for "JDK DOWNLOAD", "Server JRE DOWNLOAD", and "JRE DOWNLOAD". A red circle highlights the "JRE DOWNLOAD" button. At the bottom, a question "Which Java package do I need?" is displayed.

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

| Product / File Description | File Size | Download |
|----------------------------|-----------|--|
| Linux x86 | 54.86 MB | jre-8u111-linux-i586.rpm |
| Linux x86 | 70.65 MB | jre-8u111-linux-i586.tar.gz |
| Linux x64 | 52.75 MB | jre-8u111-linux-x64.rpm |
| Linux x64 | 68.57 MB | jre-8u111-linux-x64.tar.gz |
| Mac OS X | 64.33 MB | jre-8u111-macosx-x64.dmg |
| Mac OS X | 56 MB | jre-8u111-macosx-x64.tar.gz |
| Solaris SPARC 64-bit | 46.04 MB | jre-8u111-solaris-sparcv9.tar.gz |
| Solaris x64 | 49.88 MB | jre-8u111-solaris-x64.tar.gz |
| Windows x86 Online | 0.7 MB | jre-8u111-windows-i586-ifwt.exe |
| Windows x86 Offline | 53.53 MB | jre-8u111-windows-i586.exe |
| Windows x86 | 59.43 MB | jre-8u111-windows-i586.tar.gz |
| Windows x64 Offline | 60.31 MB | jre-8u111-windows-x64.exe |
| Windows x64 | 62.78 MB | jre-8u111-windows-x64.tar.gz |

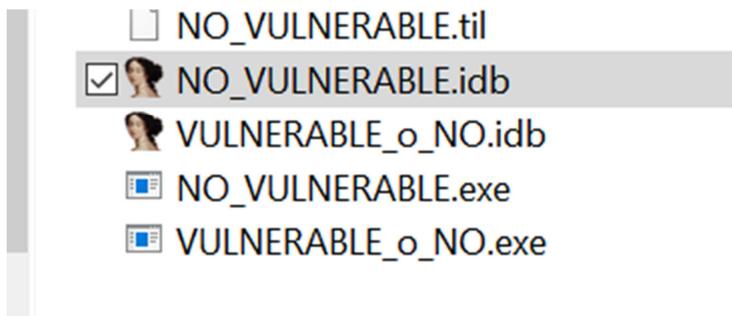
Java SE Runtime Environment 8u111

It seems to be the version downloaded and installed.

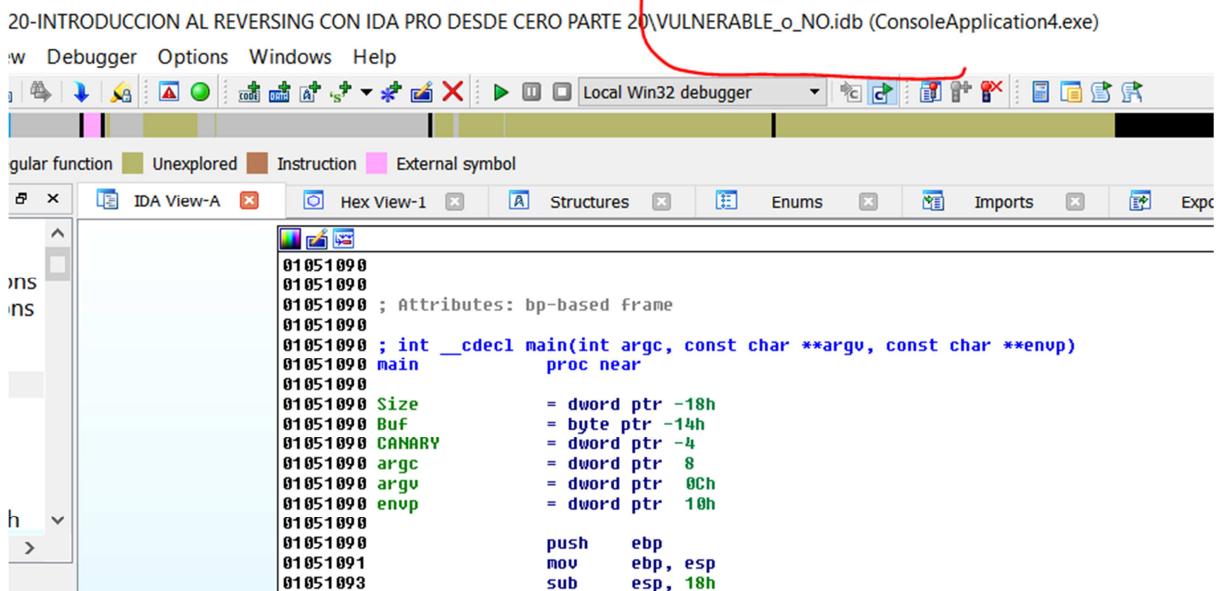


I then retry installing the bindiff and it looks like no problem, no complains.

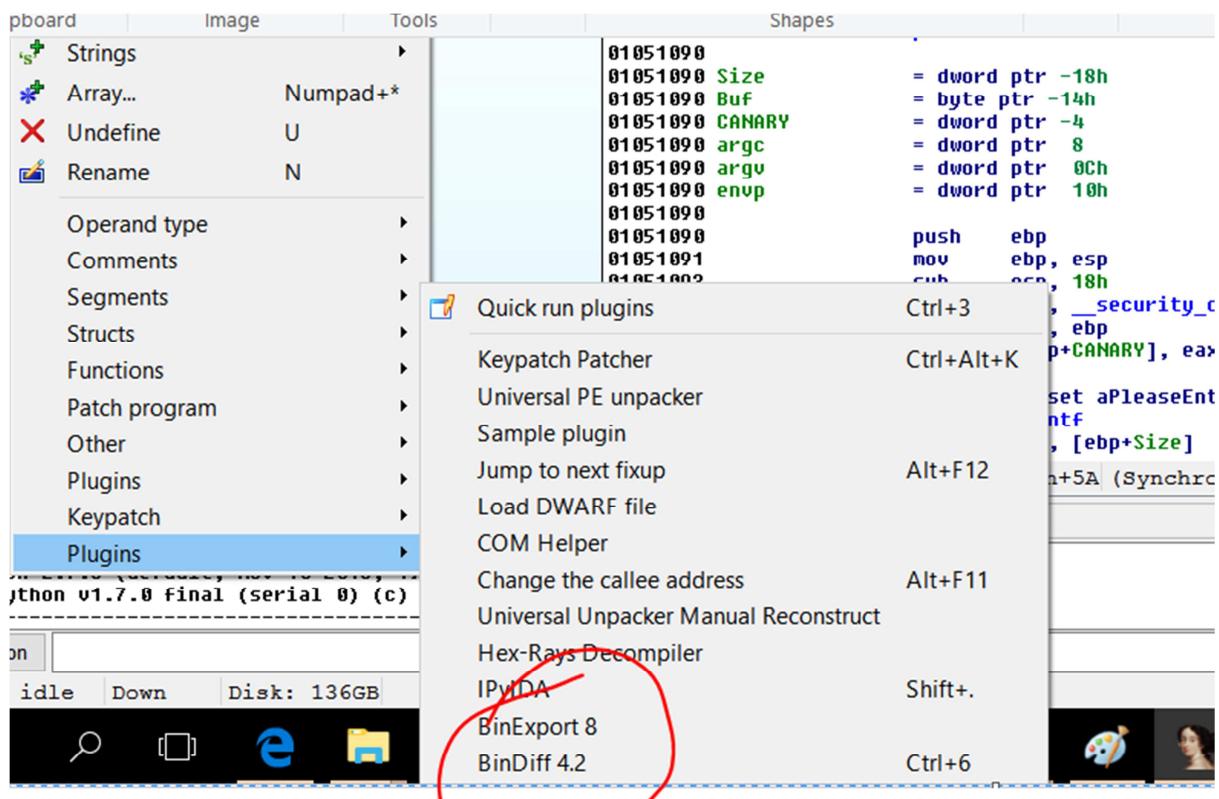
I still have the previous exercise executables vulnerable and not vulnerable, I open the new one the non-vulnerable or patched in the IDA LOADER for it to create the IDB if I did not before.



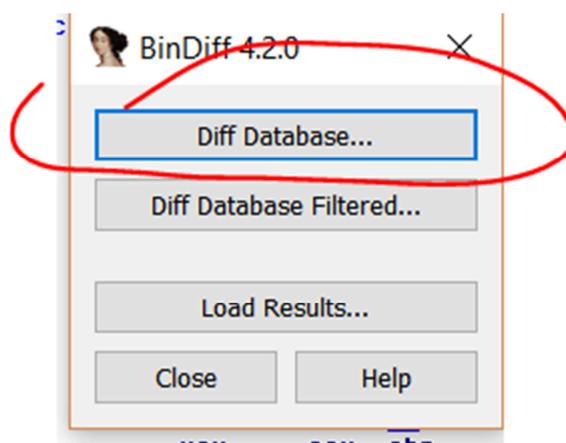
Then I open the vulnerable.



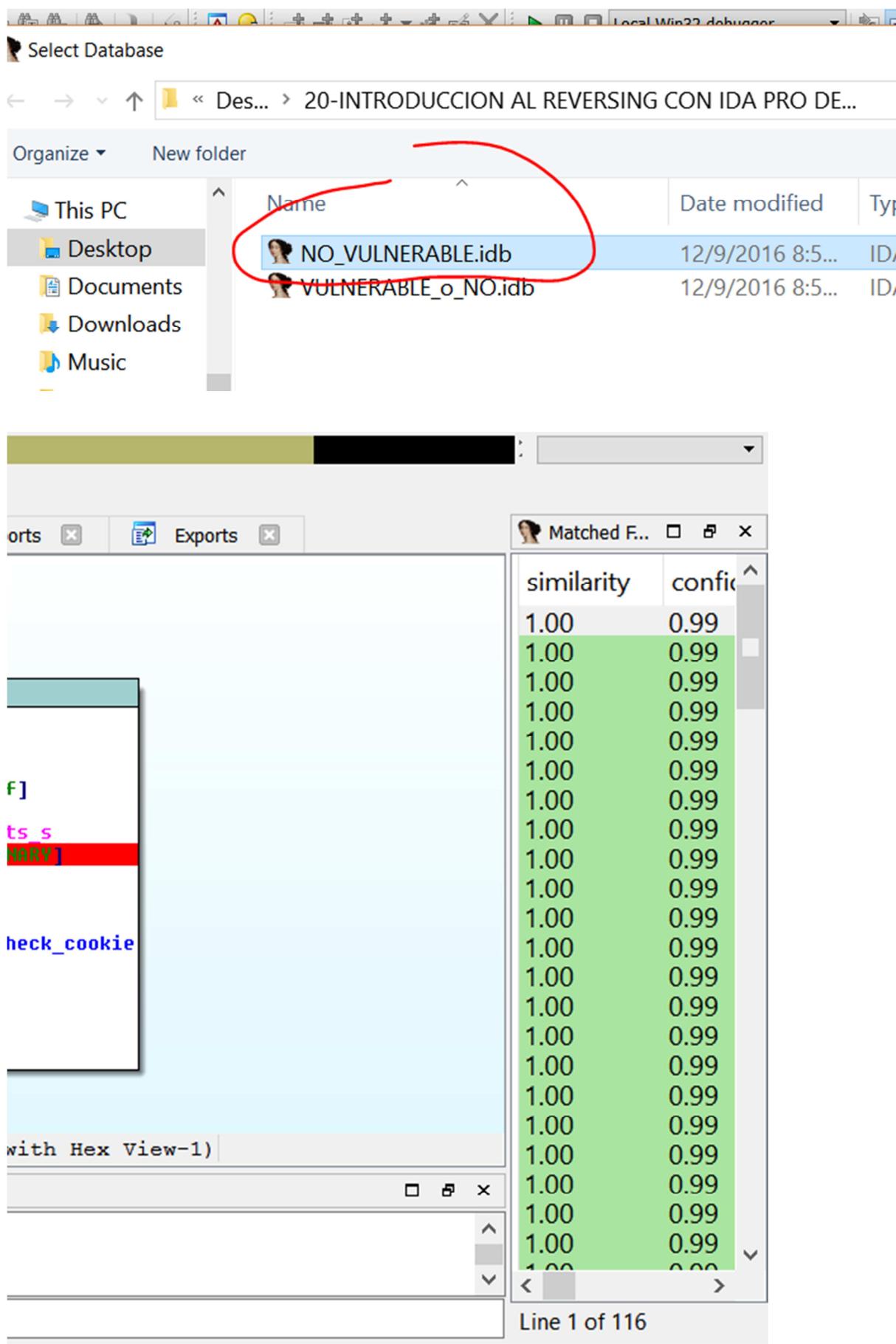
Then, I go to edit-plugins.



And I choose: BINDIFF



And I press DIFF DATABASE and look for the patched version to compare with it.



Once it is finished it shows me the results. Sometimes, it is not comfortable to see it in a column. So, I can drag it and drop it on the tab bar.

| Name | similarity | confide | change | EA primary | name primary | EA secondary |
|------|------------|---------|--------|------------|--------------------------------------|--------------|
| | 1.00 | 0.99 | ----- | 010510FD | security check cookie | 00AE10FD |
| | 1.00 | 0.99 | ----- | 0105110E | pre c initialization | 00AE110E |
| | 1.00 | 0.99 | ----- | 010511BA | pre cpp initialization | 00AE11BA |
| | 1.00 | 0.99 | ----- | 010511CC | scrt common main seh | 00AE11CC |
| | 1.00 | 0.99 | ----- | 0105133B | mainCRTStartup | 00AE133B |
| | 1.00 | 0.99 | ----- | 0105136D | report qsfailure | 00AE136D |
| | 1.00 | 0.99 | ----- | 01051468 | find pe section | 00AE1468 |
| | 1.00 | 0.99 | ----- | 010514AC | scrt acquire startup lock | 00AE14AC |
| | 1.00 | 0.99 | ----- | 010514E1 | scrt initialize crt | 00AE14E1 |
| | 1.00 | 0.99 | ----- | 0105151A | scrt initialize onexit tables | 00AE151A |
| | 1.00 | 0.99 | ----- | 010515B1 | scrt is nonwritable in current image | 00AE15B1 |
| | 1.00 | 0.99 | ----- | 0105163B | scrt release startup lock | 00AE163B |
| | 1.00 | 0.99 | ----- | 01051658 | scrt uninitialized crt | 00AE1658 |
| | 1.00 | 0.99 | ----- | 01051680 | onexit | 00AE1680 |
| | 1.00 | 0.99 | ----- | 010516BB | atexit | 00AE16BB |
| | 1.00 | 0.99 | ----- | 010516D0 | security init cookie | 00AE16D0 |
| | 1.00 | 0.99 | ----- | 010517B2 | initialize default precision | 00AE17B2 |

The first column shows the similarity. Those that say 1.00 are the same and the lower the number, the more different they are. It is convenient to click on the top of that column to order them from more different to more similar.

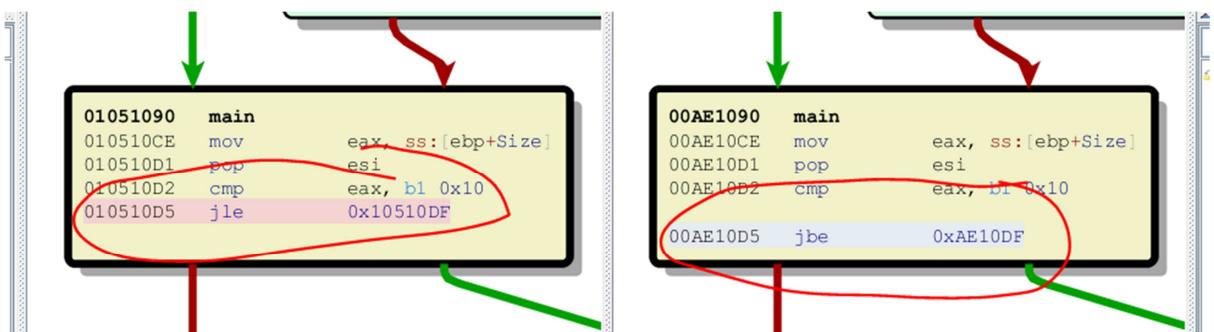
| Name | similarity | confide | change | EA primary | name primary | EA secondary |
|------|------------|---------|---------|------------|----------------------------------|--------------|
| | 0.97 | 0.98 | -I-J--- | 01051090 | main | 00AE1090 |
| | 1.00 | 0.97 | ----- | 01051CB6 | scrt stub for acrt uninitialized | 00AE1CB6 |
| | 1.00 | 0.97 | ----- | 01051C80 | IsProcessorFeaturePresent(x) | 00AE1CB0 |
| | 1.00 | 0.97 | ----- | 01051CAA | terminate | 00AE1CAA |
| | 1.00 | 0.97 | ----- | 01051CA4 | controlfp s | 00AE1CA4 |
| | 1.00 | 0.97 | ----- | 01051C9E | crt atexit | 00AE1C9E |
| | 1.00 | 0.97 | ----- | 01051C98 | register onexit function | 00AE1C98 |
| | 1.00 | 0.97 | ----- | 01051C92 | initialize onexit table | 00AE1C92 |
| | 1.00 | 0.97 | ----- | 01051C8C | p commode | 00AE1C8C |
| | 1.00 | 0.97 | ----- | 01051C86 | set new mode | 00AE1C86 |

We see that there is only one with similarity smaller than 1.

| Name | similarity | confide | change | EA primary | name primary | |
|------|------------|---------|---------|------------|---|----------------|
| | 0.97 | 0.98 | -I-J--- | 01051090 | main | |
| | 1.00 | 0.99 | ----- | 010510FD | Delete Match | Del |
| | 1.00 | 0.99 | ----- | 0105110E | View Flowgraphs | Ctrl+E |
| | 1.00 | 0.99 | ----- | 010511BA | Copy | Ctrl+C |
| | 1.00 | 0.99 | ----- | 010511CC | Copy all | Ctrl+Shift+Ins |
| | 1.00 | 0.99 | ----- | 0105133B | Unsort | |
| | 1.00 | 0.99 | ----- | 0105136D | Quick filter | Ctrl+F |
| | 1.00 | 0.99 | ----- | 01051468 | Modify filters... | Ctrl+Shift+F |
| | 1.00 | 0.99 | ----- | 010514AC | Import Symbols and Comments | |
| | 1.00 | 0.99 | ----- | 010514E1 | Import Symbols and Comments as external lib | |
| | 1.00 | 0.99 | ----- | 0105151A | | |
| | 1.00 | 0.99 | ----- | 010515B1 | | |
| | 1.00 | 0.99 | ----- | 0105163B | | |

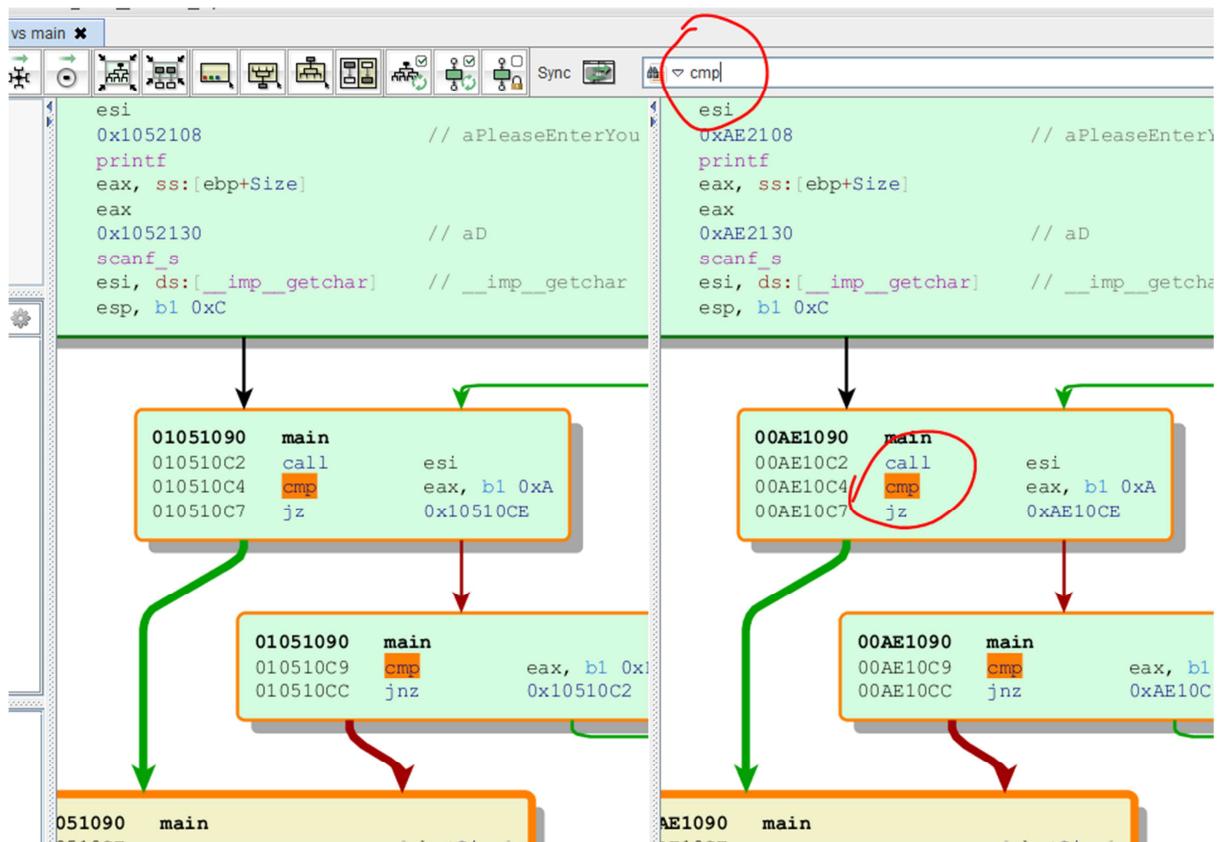


The green blocks are similar. The yellow blocks have some change and the red or gray ones are added.

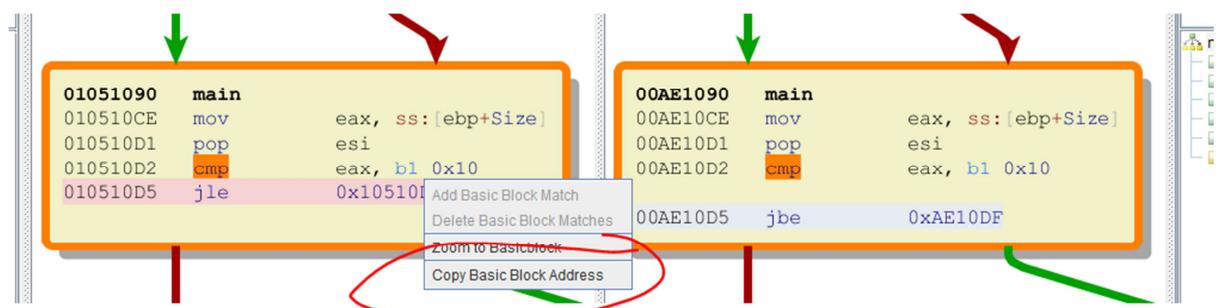


There we see the change. As we know to change a JLE jump by JBE, it is something that can avoid a BUFFER OVERFLOW, therefore if in a program of which we have the vulnerable version and the patched version and looking at the changed functions in some of them we find this we will know that we will have to reverse that function statically to see if it really is the vulnerable of the program.

One of the advantages of BINDIFF over the other two is that the graph is interactive, not an image only, it has a search engine above.

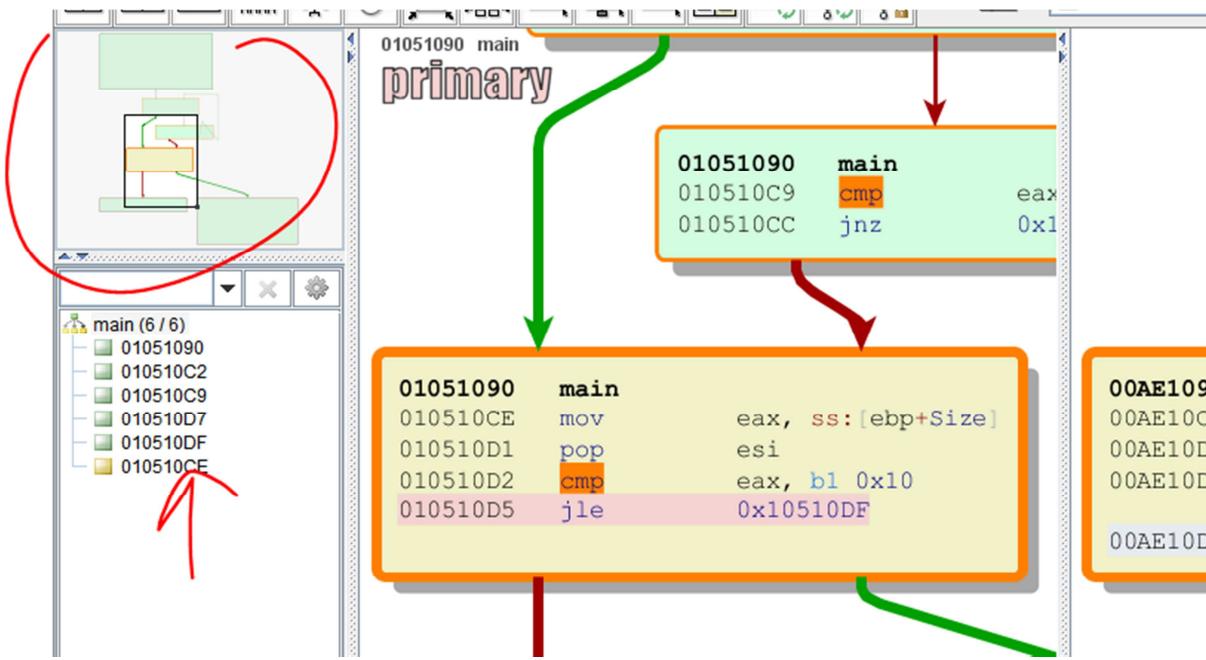


This is very useful, and you can also find addresses and any text that is on the chart.



We can copy the address of the block to paste it in IDA and go there.

We also have a graphical browser to navigate the function and list of blocks.



We can mark a block and in the menu we have **select ancestor** or **select successors** so that we obscure the blocks of the path inside the function to reach the same initial block, or from there, in this case is a simple function, but in large and complex finding the way to a block is very important.

There are many good things about the Bindiff in the graphic section, it has some problems of match in big programs, but it is one of the best options to take into account.

TURBODIFF

It is a plw created by my colleague Nicolas Economou from Core, it will be attached with the tutor, it can also be downloaded from the website of CORE SECURITY but it is an earlier version than the one I attached. The PLW is copied to the plugins folder inside the IDA installation folder.

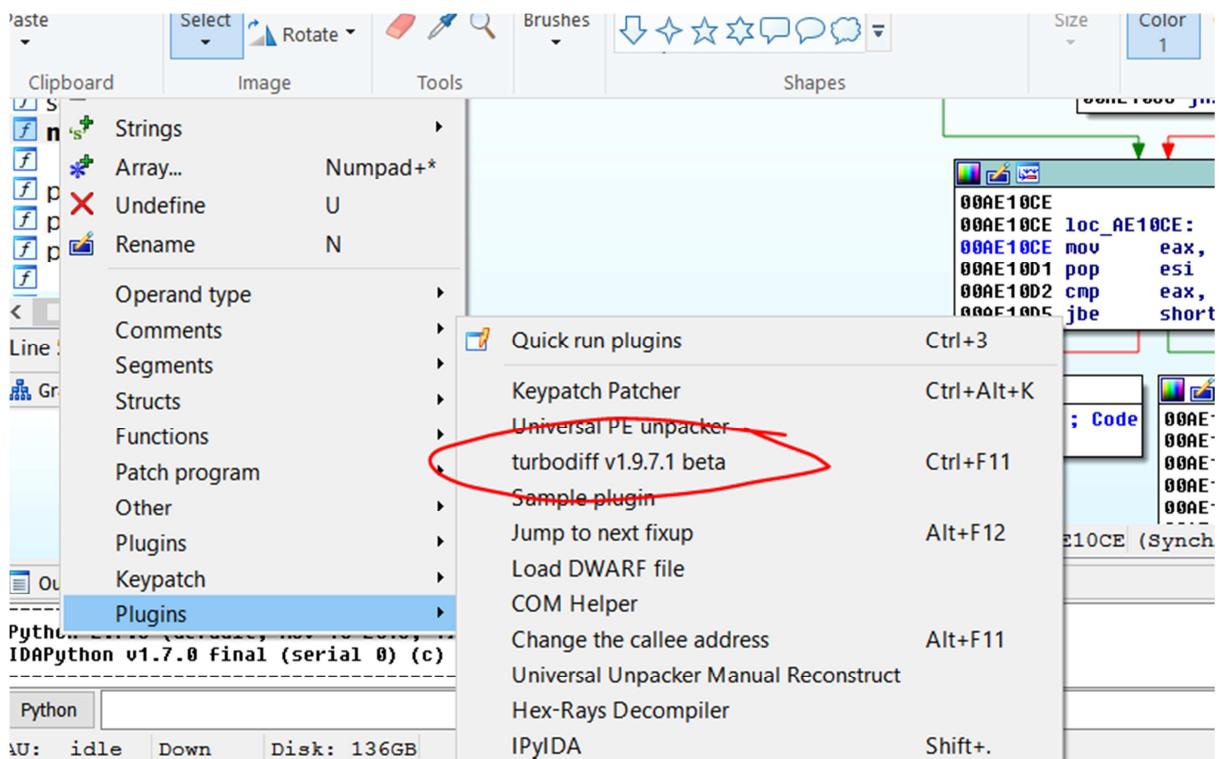
| | Name | Date modified | Type |
|---|-------------|---------------------|-------------|
| □ | ida.int | 4/13/2015 6:35 P... | INT FILE |
| □ | ida64.int | 4/13/2015 6:35 P... | INT File |
| □ | idacolor.cf | 4/13/2015 6:35 P... | CF File |
| □ | idaw.exe | 4/13/2015 6:35 P... | Application |
| □ | idaw64.exe | 4/13/2015 6:35 P... | Application |
| □ | symsrv.dll | 4/13/2015 6:35 P... | Application |
| □ | python | 12/9/2016 9:11 ... | File folder |
| □ | plugins | 12/9/2016 8:54 ... | File folder |
| □ | til | 9/12/2016 9:46 P... | File folder |
| □ | sig | 9/12/2016 9:46 P... | File folder |
| □ | procs | 9/12/2016 9:46 P... | File folder |
| □ | loaders | 9/12/2016 9:46 P... | File folder |
| □ | dbgsrv | 9/12/2016 9:46 P... | File folder |
| □ | idc | 9/12/2016 9:46 P... | File folder |
| □ | ids | 9/12/2016 9:46 P... | File folder |
| □ | cfg | 9/12/2016 9:46 P... | File folder |

| 3.10 MB

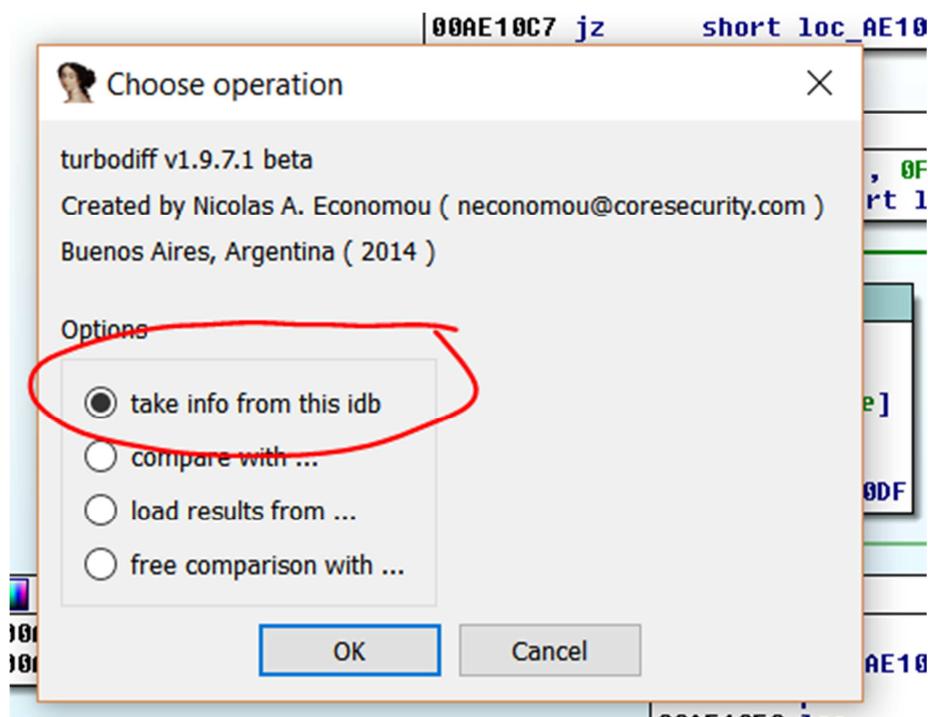
| | Name | Date modified | Type | Size |
|-------------------------------------|---------------|------------------|----------|------|
| □ | strings.plw | 4/13/2015 6:3... | PLW File | / |
| □ | tds.p64 | 4/13/2015 6:3... | P64 File | 19 |
| □ | tds.plw | 4/13/2015 6:3... | PLW File | 18 |
| <input checked="" type="checkbox"/> | turbodiff.plw | 5/29/2001 9:1... | PLW File | 117 |
| □ | uiswitch.p64 | 4/13/2015 6:3... | P64 File | 13 |
| □ | uiswitch.plw | 4/13/2015 6:3... | PLW File | 13 |

I will have to restart the IDA to load it.

As always I load the non-VULNERABLE version first.



It is necessary to take the information of each idb to be compared.



So I do that in this first one.

The screenshot shows the IDA Pro interface with the 'Output window' tab selected. The window displays the following text:

```
analyzing 95%...
analyzing 97%...
analyzing 98%...
analyzing 100%...
collapsing ae19de --> ae17a3
generating C:\Users\ricna\Desktop\20-INTRODUCCION AL REVERSING CON IDA PRO DESDE CERO PARTE 20\NO_VULNERABLE.dis
generating C:\Users\ricna\Desktop\20-INTRODUCCION AL REVERSING CON IDA PRO DESDE CERO PARTE 20\NO_VULNERABLE.ana
elapsed time: 0.344 sec.
done
```

A red circle highlights the 'done' message at the bottom of the output window.

Then I open the vulnerable and do the same.

The screenshot shows the IDA Pro interface with the 'Output window' tab selected. The window displays the following text:

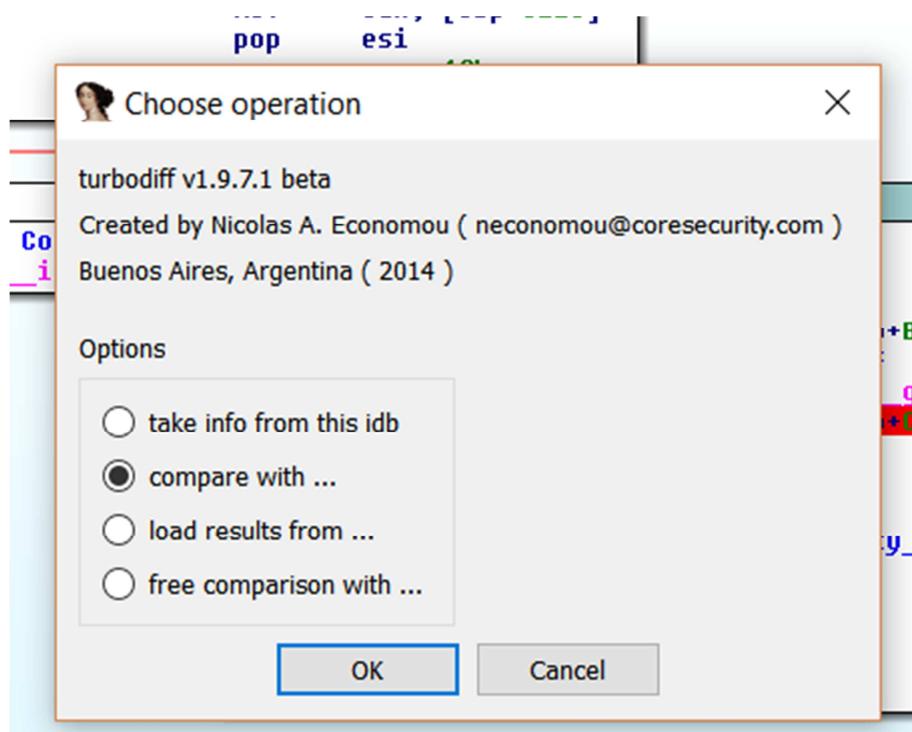
```
pre cpp initialization
scrt common main seh
Line 5 of 73
Graph overview
010510EA mov ecx, [ebp+CARRY]
010510ED add esp, 8
010510F0 xor ecx, ebp
010510F2 xor eax, eax
010510F4 call _security_check_cool
010510F9 mov esp, ebp
010510FB pop ebp
010510FC retn
010510FC main endp
010510FC

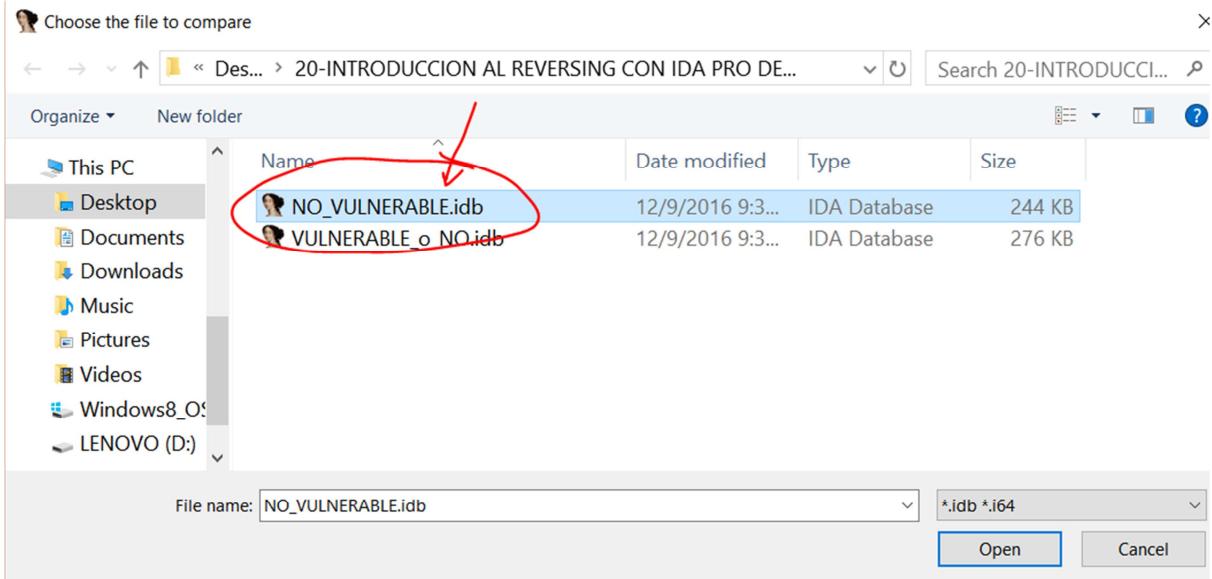
100.00% (116, 772) | 000004EA 010510EA: main+5A (Synchronized with Hex View-1)

generating C:\Users\ricna\Desktop\20-INTRODUCCION AL REVERSING CON IDA PRO DESDE CERO PARTE 20\VULNERABLE_o_NO.dis
generating C:\Users\ricna\Desktop\20-INTRODUCCION AL REVERSING CON IDA PRO DESDE CERO PARTE 20\VULNERABLE_o_NO.ana
elapsed time: 0.266 sec.
done
```

A red circle highlights the 'done' message at the bottom of the output window.

Then from the vulnerable I call the plugin again.





I look for the non-vulnerable version and accept the options that it brings by default.

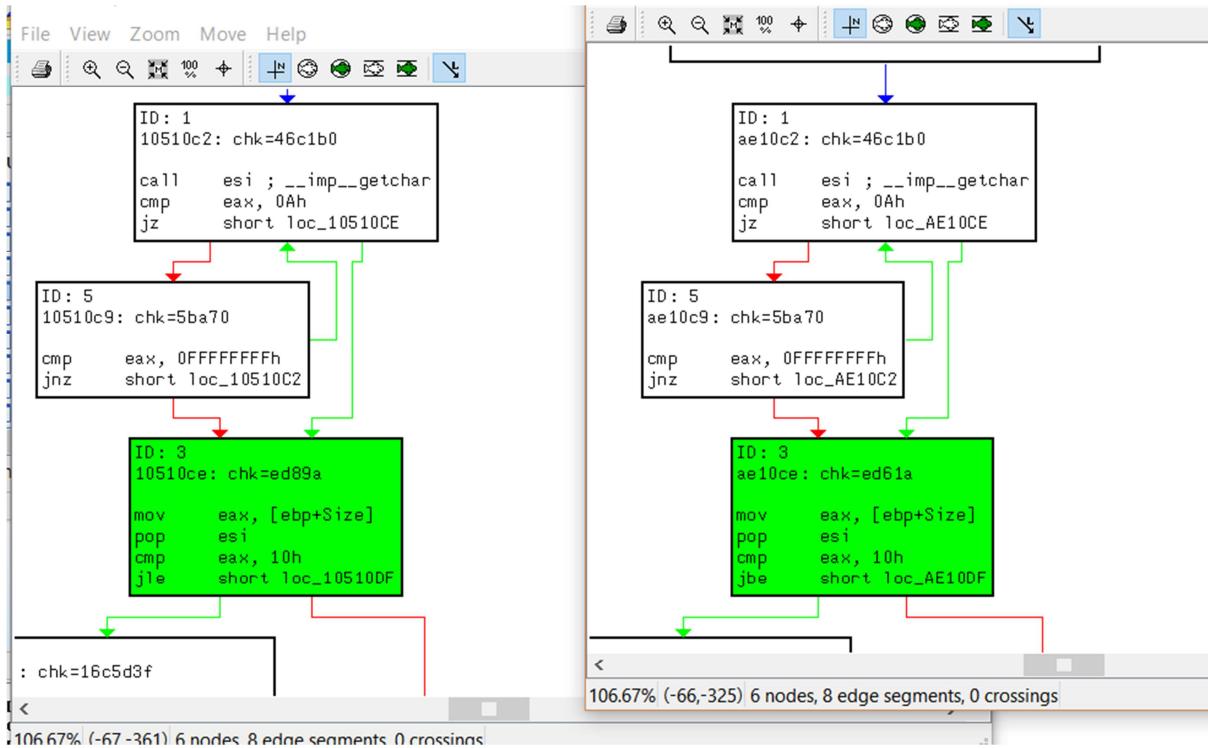
| vie category | address | name | address |
|--------------|---------|--------------------------------------|---------|
| identical | 1051000 | local stdio printf options | ae1000 |
| identical | 1051010 | local stdio scanf options | ae1010 |
| identical | 1051020 | printf | ae1020 |
| identical | 1051050 | scanf s | ae1050 |
| identical | 10510fd | security check cookie | ae10fd |
| identical | 105110e | pre c initialization | ae110e |
| identical | 10511b2 | post pgo initialization | ae11b2 |
| identical | 10511ba | pre cpp initialization | ae11ba |
| identical | 10511cc | scrt common main seh | ae11cc |
| identical | 105133b | mainCRTStartup | ae133b |
| identical | 1051345 | raise securityfailure | ae1345 |
| identical | 105136d | report qsfailure | ae136d |
| identical | 1051468 | find pe section | ae1468 |
| identical | 10514ac | scrt acquire startup lock | ae14ac |
| identical | 10514e1 | scrt initialize crt | ae14e1 |
| identical | 105151a | scrt initialize onexit tables | ae151a |
| identical | 10515b1 | scrt is nonwritable in current image | ae15b1 |

Line 1 of 72

There I can press CTRL + F and look for **changed** or **suspicious** to show me the changed ones.

| | | | | |
|---|--------------|---------|---|--------|
| S | identical | 1051cb8 | p_arqv | ae1cb8 |
| | identical | 1051c6e | cexit | ae1c6e |
| | identical | 1051c74 | c_exit | ae1c74 |
| | identical | 1051c7a | register thread local exe atexit callback | ae1c7a |
| | identical | 1051c80 | configthreadlocale | ae1c80 |
| | identical | 1051c86 | set new mode | ae1c86 |
| | identical | 1051c8c | p_commode | ae1c8c |
| | identical | 1051c92 | initialize onexit table | ae1c92 |
| | identical | 1051c98 | register onexit function | ae1c98 |
| | identical | 1051c9e | crt atexit | ae1c9e |
| | identical | 1051ca4 | controlfp s | ae1ca4 |
| | identical | 1051caa | terminate | ae1caa |
| | identical | 1051cb0 | IsProcessorFeaturePresent(x) | ae1cb0 |
| | identical | 1051cb6 | scrt stub for acrt uninitialized | ae1cb6 |
| | suspicious + | 1051090 | main | ae1090 |

There it is. Double click.



There, you will see the changed ones. There is also a color code according to the exchange rate, green for the blocks with minimum changes, yellow for the very changed blocks and red for the added blocks. Obviously, the graphics are pictures and they are not interactive, but it is a very fast Differ. It really is the fastest. It is very noticeable in large executables and does not show too many stupid changes like Bindiff, assuming many as not important which in big works is appreciated. If one does not like the shape of the graphics can use both differs at a time and then see the results in the graph of the Bindiff.

DIAPHORA

The Diaphora is a plugin made in python by Joxean Koret.

<https://github.com/joxeankoret/diaphora>

joxeankoret / diaphora

Code Issues 22 Pull requests 0 Projects 0 Pulse Graphs

Diaphora, a Free and Open Source program diffing tool <http://diaphora.re>

114 commits 2 branches 0 releases 7 contributors GPL-2.0

Branch: master New pull request

joxeankoret committed on GitHub Merge pull request #67 from jarnovanleeuwen/master ...

doc Added documentation for the heuristics Latest commit 4a56560 on 11 Oct 8 months ago

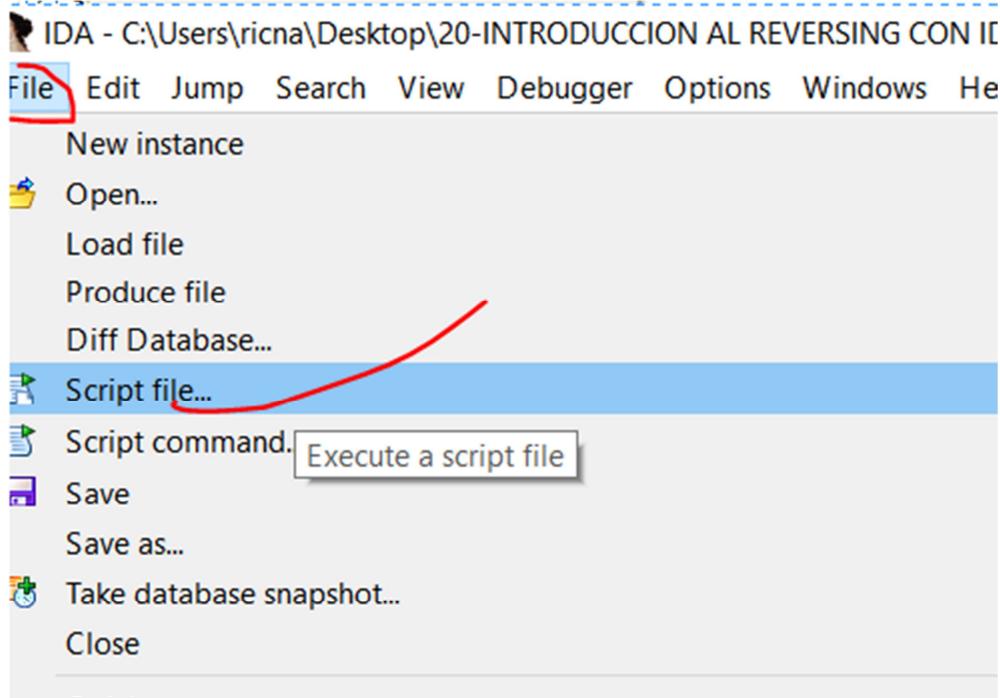
Find file Clone or download

It is not necessary to install it. I can decompress it anywhere and it is only necessary to have Python installed on the machine which if we have IDA we will have it.

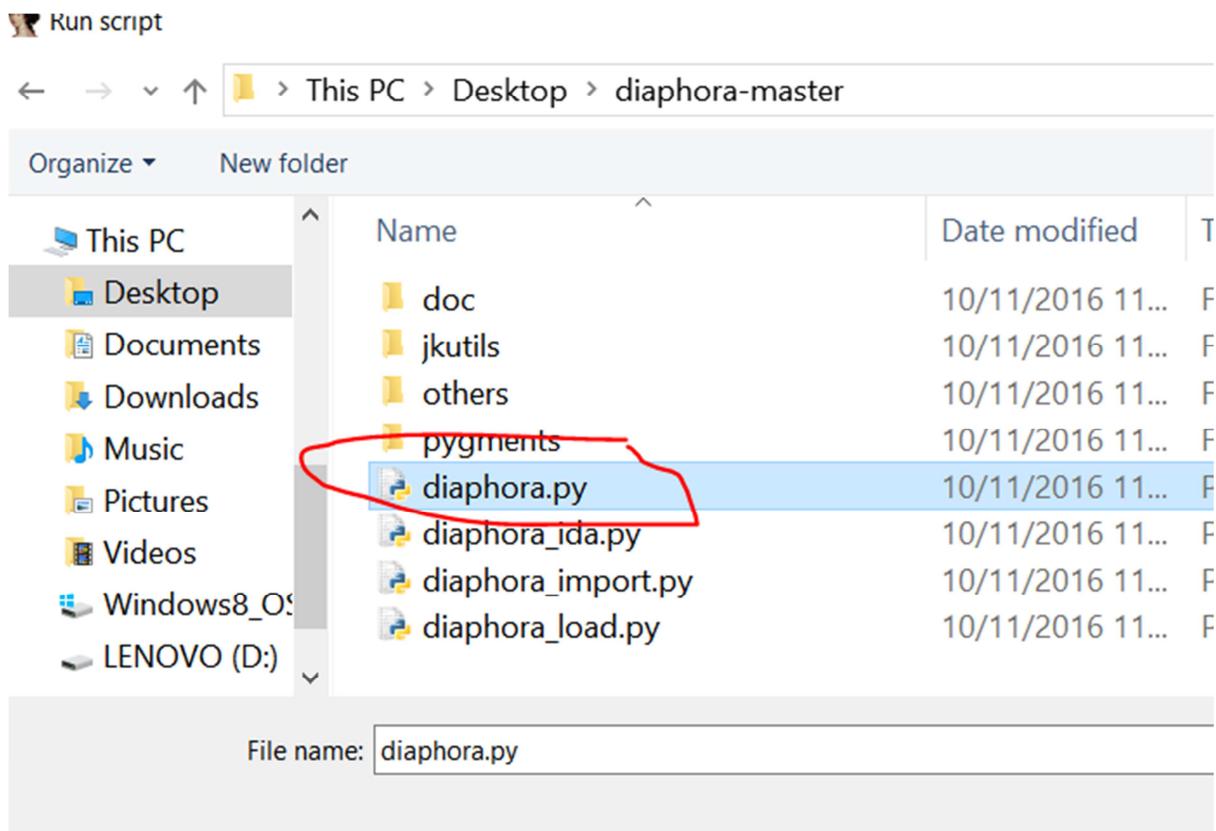
| Copy | Paste | Paste shortcut | Move to | Copy to | Delete | Rename | New folder | Properties | History | Invert selection | Select |
|------------------------|--------|----------------|---------|-----------|---------|-------------|-------------|---------------|---------------|------------------|--------|
| Clipboard | | | | Organize | | | | | | | |
| diaphora-master | | | | | | | | | | | |
| ck access | esktop | ownloads | objects | documents | ictures | oogle Drive | I-INTRODUCC | onsoleApplica | ew folder (6) | CKED_PRAC | eDrive |

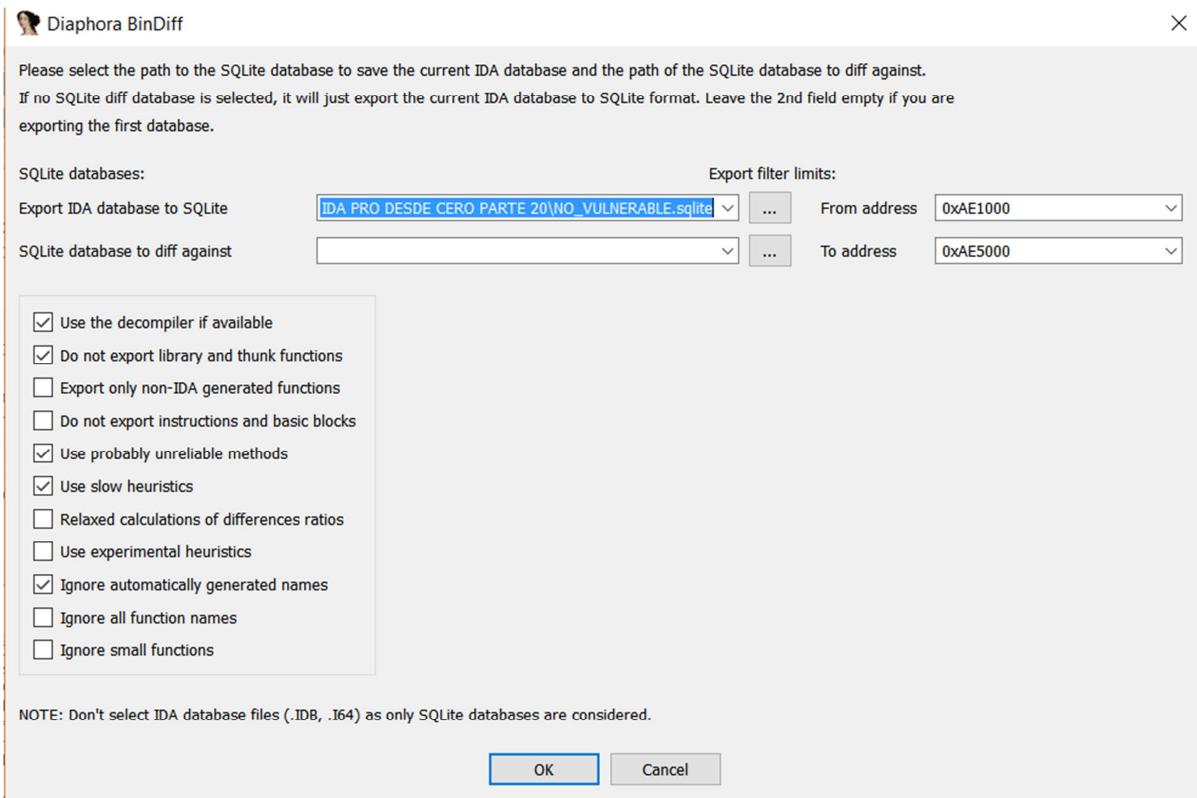
| Name | Date modified | Type | Size |
|--------------------|------------------|----------------|-------|
| doc | 10/11/2016 11... | File folder | |
| jkutils | 10/11/2016 11... | File folder | |
| others | 10/11/2016 11... | File folder | |
| pygments | 10/11/2016 11... | File folder | |
| .gitignore | 10/11/2016 11... | GITIGNORE File | 1 KB |
| diaphora.py | 10/11/2016 11... | Python File | 90 KB |
| diaphora_ida.py | 10/11/2016 11... | Python File | 60 KB |
| diaphora_import.py | 10/11/2016 11... | Python File | 2 KB |
| diaphora_load.py | 10/11/2016 11... | Python File | 2 KB |
| LICENSE | 10/11/2016 11... | File | 18 KB |
| README.md | 10/11/2016 11... | MD File | 3 KB |

So we will go as usual. First the patch or not vulnerable in IDA.

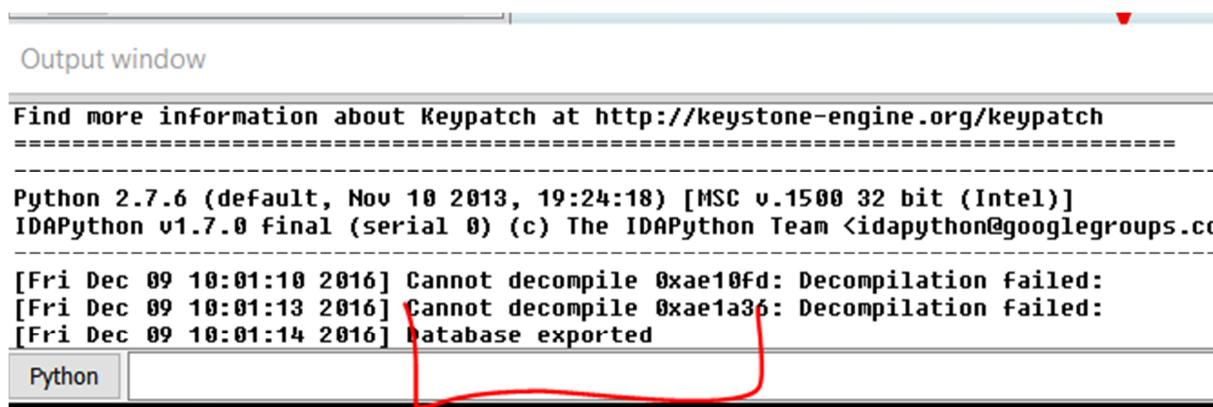


In FILE-SCRIPT FILE open the search engine and go where we decompressed the Diaphora and look for **diaphora.py**.

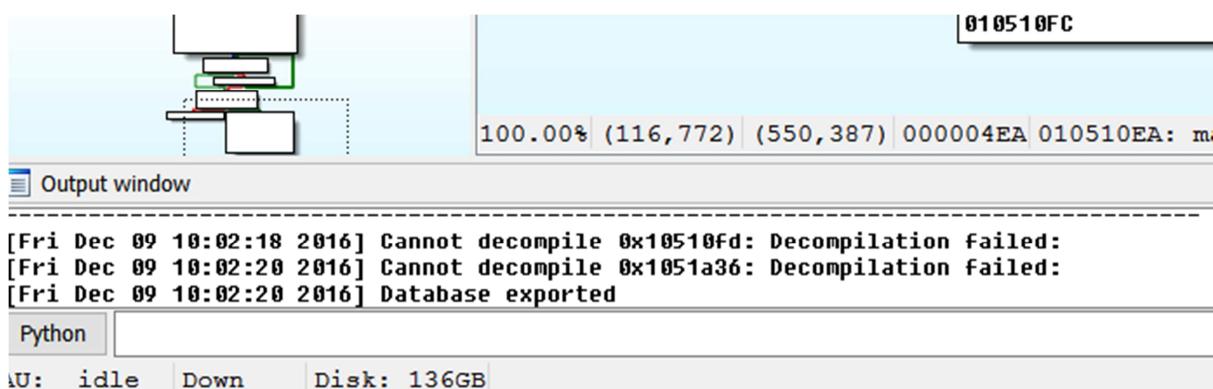




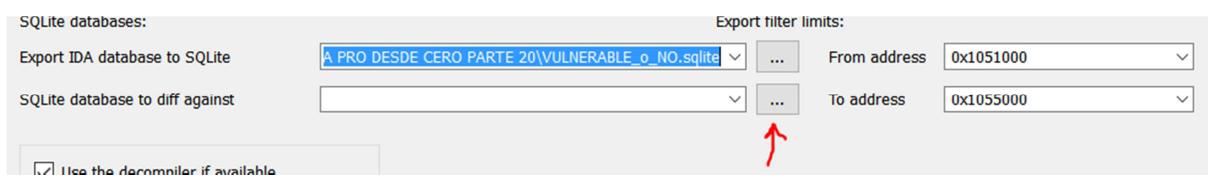
As it comes, we press OK to export the database to SQL.



When it finishes we open the vulnerable in IDA and we do the same we open diaphora.py and without changing anything we export the database.

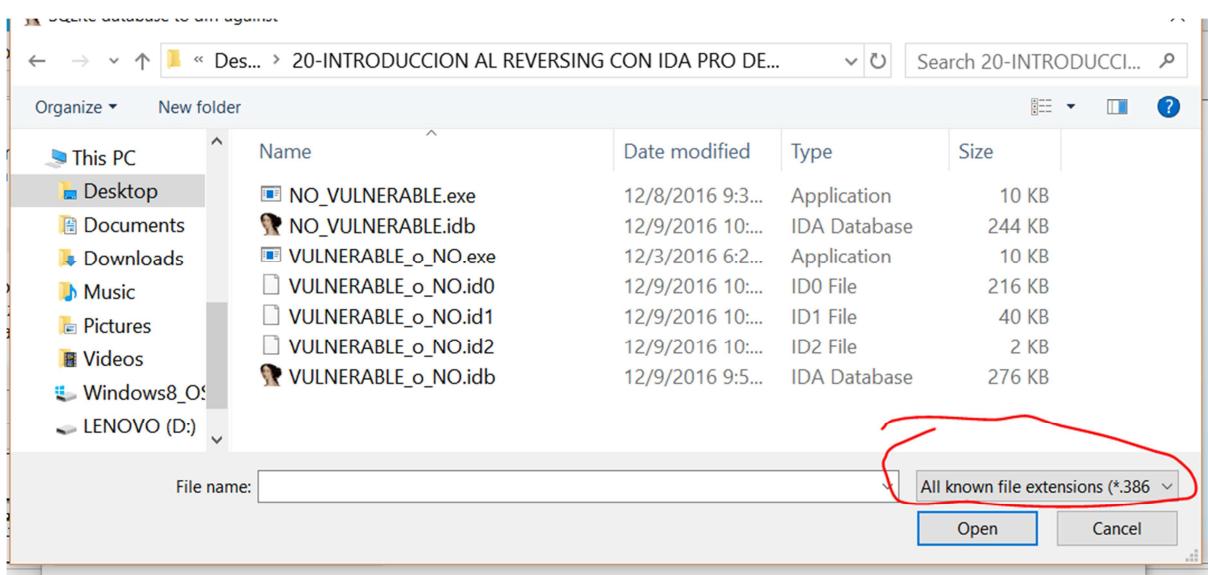


Once we did the same in both, we reopen the diaphora.py in the vulnerable version but this time.

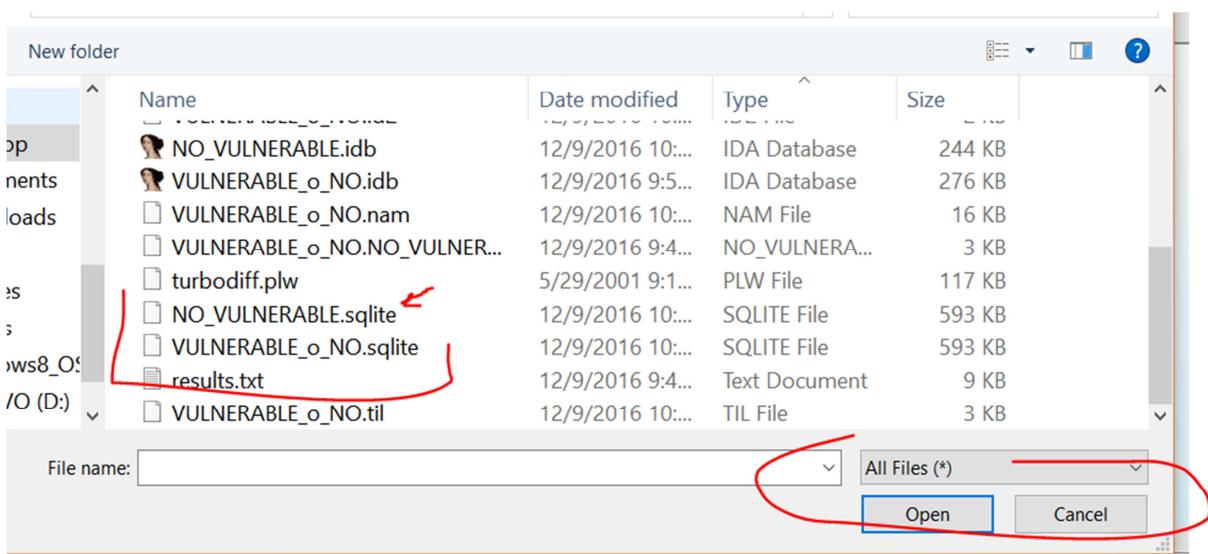


In the second place we look for the SQL database of the patching that it exported before.

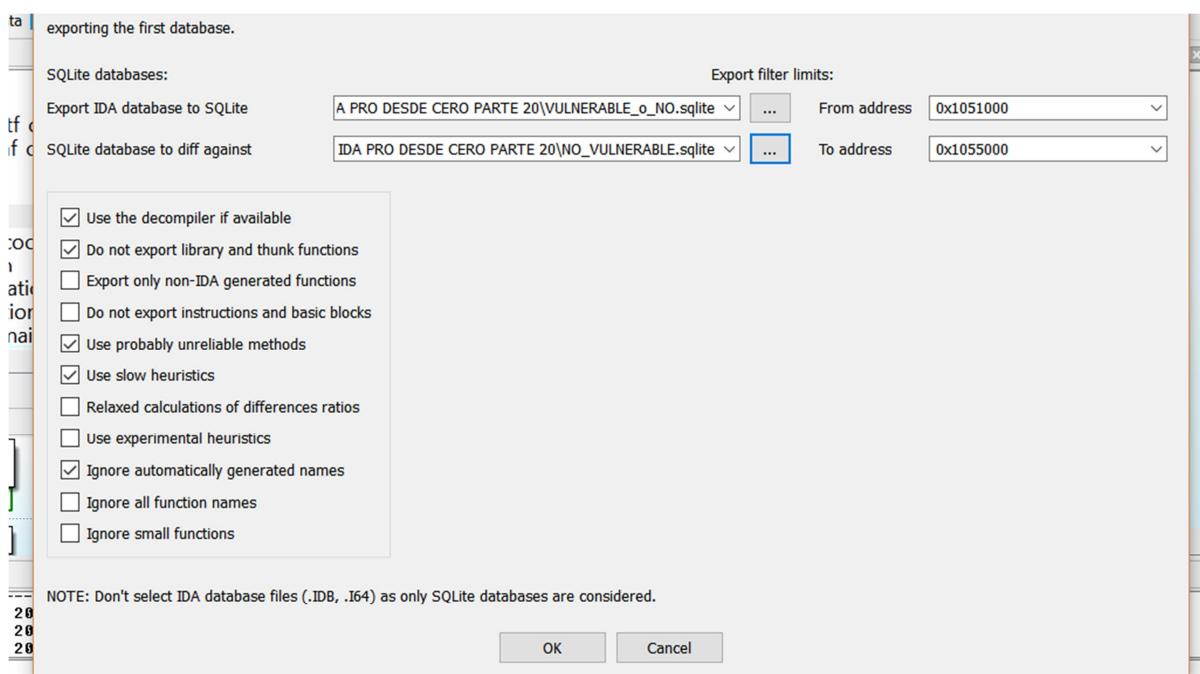
We see that when we go to the folder, there seems to be nothing



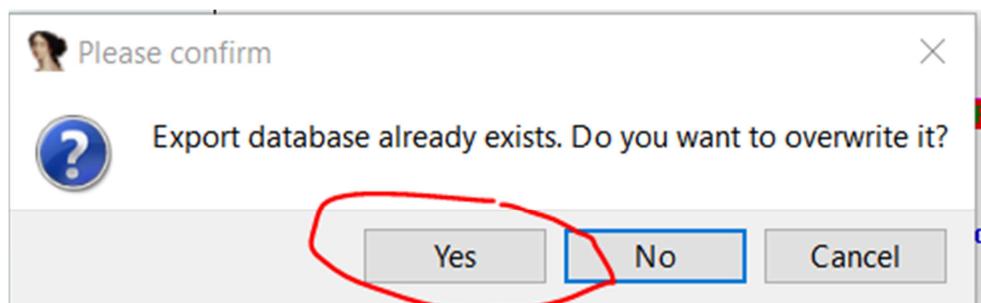
But it's because of the file filter. We change it to show everything.



And we look for the non-vulnerable.



I press OK, just like this.



We see that there is a BEST MATCHES tab with which there is no doubt that they are the same.

The screenshot shows the IDA Pro interface with the 'Best matches' tab selected. The window displays a table with columns: Name, Address, Name, Address 2, Name 2, and Ratio. The data in the table is as follows:

| Name | Address | Name | Address 2 | Name 2 | Ratio |
|------|----------|---------------------------|-----------|------------------------------|-------|
| 000 | 01051020 | printf | 00ae1020 | printf | 1.000 |
| 001 | 01051050 | scanf s | 00ae1050 | scanf s | 1.000 |
| 002 | 010510fd | security check cookie | 00ae10fd | security check cookie | 1.000 |
| 003 | 010511b2 | post pgo initialization | 00ae11b2 | post pgo initialization | 1.000 |
| 004 | 010511ba | pre cpp initialization | 00ae11ba | pre cpp initialization | 1.000 |
| 005 | 01051345 | raise securityfailure | 00ae1345 | raise securityfailure | 1.000 |
| 006 | 01051468 | find pe section | 00ae1468 | find pe section | 1.000 |
| 007 | 010514e1 | scrt initialize crt | 00ae14e1 | scrt initialize crt | 1.000 |
| 008 | 01051658 | scrt uninitialized crt | 00ae1658 | scrt uninitialized crt | 1.000 |
| 009 | 010516bb | atexit | 00ae16bb | atexit | 1.000 |
| 010 | 0105176c | get startup arqv mode | 00ae176c | get startup arqv mode | 1.000 |
| 011 | 01051770 | get startup file mode | 00ae1770 | get startup file mode | 1.000 |
| 012 | 01051782 | initialize default precis | 00ae1782 | initialize default precision | 1.000 |

In the tab PARTIAL MATCHES we see the ones that possibly were changed.

| Line | Address | Name | Address 2 | Name 2 | Ratio | B Bloc |
|-------|----------|-------------------------|-----------|--------------------------------------|-------|--------|
| 00000 | 01051090 | main | 00ae1090 | main | 0.930 | 6 |
| 00001 | 010515b1 | scrt is nonwritable ... | 00ae15b1 | scrt is nonwritable in current image | 0.900 | 10 |

There, the version is changed. We see that it found two changes. One of the things that Diaphora has is that it is very precise. Sometimes, that's good, but sometimes when you have hundreds of functions, you want it to be a little more relaxed and not show as many nonsense as changes.

| Line | Address | Name | Address 2 | Name 2 |
|-------|----------|------------|-----------|--------|
| 00000 | 01051090 | main | | |
| 00001 | 010515b1 | scrt is no | | |

Insert... Ins
 Delete Del
 Edit... Ctrl+E
 Refresh Ctrl+U
 Copy Ctrl+C
 Copy all Ctrl+Shift+Ins
 Quick filter Ctrl+F
 Modify filters... Ctrl+Shift+F
 Diff assembly
 Diff pseudo-code
Diff assembly in a graph

Import selected
 Import *all* functions
 Import *all* data for sub_* functions
 Highlight matches
 Unhighlight matches
 Save diffing results

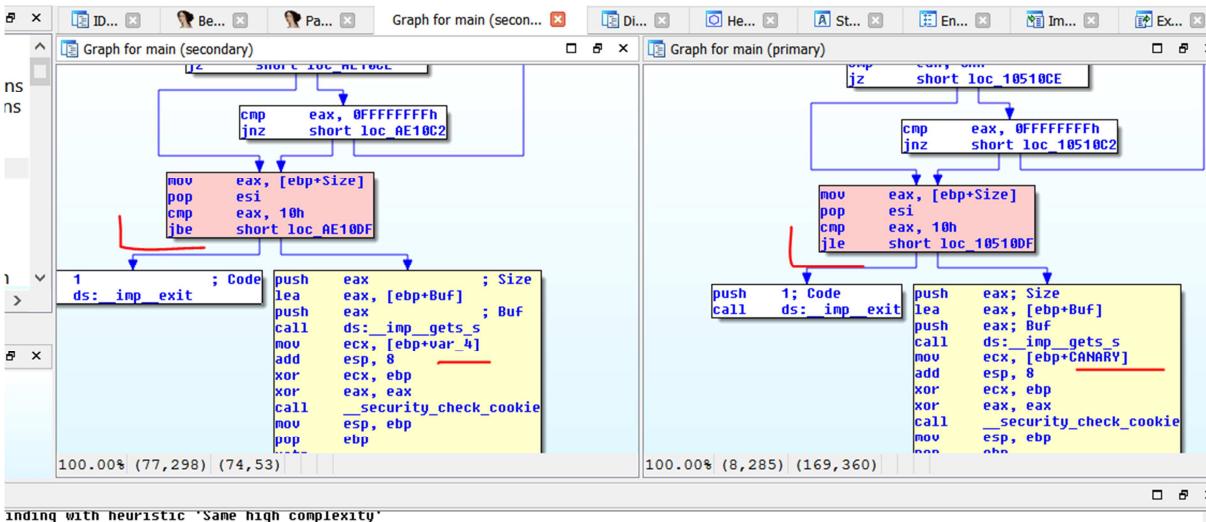
We see that the first DIFF ASSEMBLY has several options to graph.

```

FUNCTION UNEXPLORED INSTRUCTION EXTERNAL SYMBOL
IDA ... Best ... Partial ... Diff assembler main... Hex ... Str... En... Im...
1710c_10510c2: 1710c_ae10c2:
18 call esi ; _imp_getchar 18 call esi ; _imp_getchar
19 cmp eax, 0Ah 19 cmp eax, 0Ah
20 jz short loc_10510CE 20 jz short loc_AE10CE
21loc_10510c9: 21loc_ae10c9:
22 cmp eax, OFFFFFFFFh 22 cmp eax, OFFFFFFFFh
23 jnz short loc_10510C2 23 jnz short loc_AE10C2
24loc_10510ce: 24loc_ae10ce:
25 mov eax, [ebp+Size] 25 mov eax, [ebp+Size]
26 pop esi 26 pop esi
27 cmp eax, 10h 27 cmp eax, 10h
28 ble short loc_10510DF 28 jbe short loc_AE10DF
29loc_10510d7: 29loc_ae10d7:
30 push 1: Code 30 push 1 ; Code
31 call ds:_imp_exit 31 call ds:_imp_exit
32loc_10510df: 32loc_ae10df:
33 push eax; Size 33 push eax ; Size
34 lea eax, [ebp+Buf] 34 lea eax, [ebp+Buf]
35 push eax: Buf 35 push eax : Buf
36 call ds:_imp_gets_s 36 call ds:_imp_gets_s
37 mov ecx, [ebp+CANARY] 37 mov ecx, [ebp+var_4]
38 add esp, 8 38 add esp, 8
39 xor ecx, ebp 39 xor ecx, ebp

```

It's like very precise and detailed but when you see a hundred functions like this you want to die. Let's see the second option **DIFF ASSEMBLY IN A GRAPH**.



This graph is a little better, although it is not interactive, the important block changed is in red and in yellow those that have minor changes like the name of a variable.

The other **DIFF PSEUDO CODE** option uses the HEX RAYS plugin that comes with the included IDA that tries to build a source code from the executable.

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     int v3; // eax@2
4     char v5; // [sp+0h] [bp-1Ch]@0
5     rsize_t Size; // [sp+4h] [bp-18h]@1
6     char Buf[16]; // [sp+8h] [bp-14h]@6
7
8     printf((int)"\nPlease Enter Your Number of Choice: \n", v5);
9     scanf_s((int)"%d", (unsigned int)&Size);
10    do
11        v3 = _getchar();
12    while ( v3 != 10 && v3 != -1 );
13    if ( (signed int)Size > 16 )
14        _exit(1);
15    _gets_s(Buf, Size);
16    return 0;
17}

```

```

1 int __cdecl main(int argc, const char **argv, con
2 {
3     int v3; // eax@2
4     char v5; // [sp+0h] [bp-1Ch]@0
5     rsize_t Size; // [sp+4h] [bp-18h]@1
6     char Buf; // [sp+8h] [bp-14h]@6
7
8     printf((int)"\nPlease Enter Your Number of Choi
9     scanf_s((int)"%d", (unsigned int)&Size);
10    do
11        v3 = _getchar();
12    while ( v3 != 10 && v3 != -1 );
13    if ( Size > 0x10 )
14        _exit(1);
15    _gets_s(&Buf, Size);
16    return 0;
17}

```

We see that in the vulnerable that we had reversed by hand and determined that there was a buffer of 16 bytes, that variable Buf is detected as buffer, but in the other as we did not do the same work it does not detect it but as a variable char nothing more. It also shows that the variable is signed in the vulnerable and in the other it says nothing which is supposed to be unsigned. Another characteristic of Diaphora is that it is the slowest (it is programmed in Python and turbodiff in C) and in big executables the analysis and mach are very long.

I attached the IDA1.exe file. I would like to be analyzed and see if it is vulnerable and also if you can overflow the buffer and modify the flow of the

program to show us the msg of good guy. The exercise is openly discussed on both the Crackslatinos mailing list and our telegram group.

<https://telegram.me/CLSExploits>

Ricardo Narvaja

Translated by: @IvinsonCLS