CS6542 - Graduate –Computer Network Defense – Spring 2017

Lab 19-1 page 489
Due 04th April 2017, 6:00 pm.
100 points

Policy on collaboration: All examinations, papers, and other graded work products and assignments are to be completed in conformance with The George Washington University Code of Academic Integrity. Each student is expected to write his or her own HW out independently; you may not copy one another's assignments, even in part. You may not collaborate with others on the test and final.
You are expected to cite all your sources in any written work that is not closed book: papers, books, web sites, discussions with others - faculty, friends, students. For example, if, in a group, one student has a major idea that leads to a solution to a HW problem, all other students in the group should cite this student.
You may not refer to solutions to previous years' problem sets, or ask for help students from previous years. Any violations will be treated as violations of the Code of Academic Integrity.

Please work on each lab and capture screenshot of tasks along with your words and analysis of each slide. PLEASE submit all Labs on Blackboard only. Name your files:

Late submission

Please note that, there is a %10 penalty for late submission until next project due date, and also there is NO grade for project submission after the next project due date. Since we don't have any more lab, the late submission will be one full week after the actual due date.

1. Lab 19-1 Our First Diff

For this lab in particular, copy the two ELF binary files name and name2 from Lab19 tab in blackboard repository and place them in the folder C:\grayhat\app_diff\. You will need to create those subfolder.

1.1. System Requirements:

For Lab 19-1 you may wish to download the free version of IDA 6.5 and the appropriate files to get turbodiff up and running. The following instructions will walk you through this process. All files that are required to this lab are in Lab19.zip file. Please download and extract it in to your Windows 7 VM.

1.1.1. Download IDA software

Download the free version of IDA 5.0 from the Blackboard. Follow the simple installation instructions and accept all defaults.

1.1.2. Download turbodiff

The turbodiff file is located in the Lab19.zip file however you can follow the process and download by yourself.

1. Go to the Core Security, Core Labs site to download turbodiff:

   http://corelabs.coresecurity.com/index.php?module=Wiki&action=view&type=tool&name=turbodiff
   or the file is located in the lab19.zip file.

2. Go to the section of the page titled **Downloads** and download the latest stable release 1.01b_r2. This will download the file: **turbodiff_1.01b_r2_ida_free_5.rar**, which you must extract with a tool such as 7zip.
3. Once you extract this file, there are two files you must copy to your IDA Free folder under **Program Files** or **Program Files (x86)** if you are on a 64-bit version of Windows.

1.1.3. Copy turbodiff required files to IDA directory

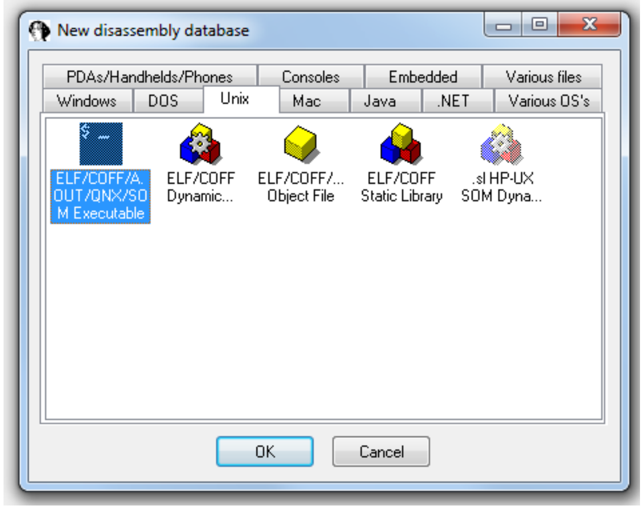
1. Navigate to Lab19\turbodiff_1.01b_r2_ida_free_5\
2. The first file is turbodiff.cfg which must be copied to: C:\Program Files (x86)\IDA Free\cfg\.
3. The second file is turbodiff.plw which must be copied to: C:\Program Files (x86)\IDA Free\plugins\.

Once you have completed above steps, turbodiff should be successfully installed.

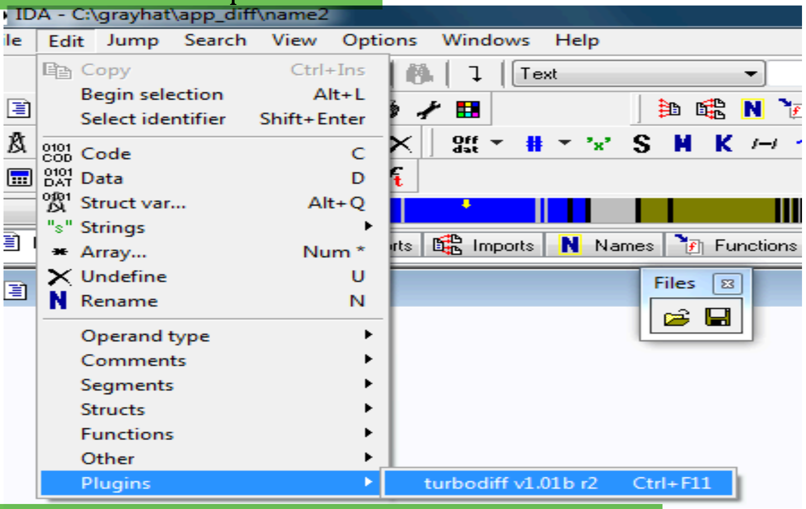
1.2. Tasks:

In this lab, you will perform a simple diff against the code previously shown in the "Application Diffing" section. The ELF binary files **name** and **name2** are to be compared. The **name** file is the unpatched one and **name2** is the patched one.

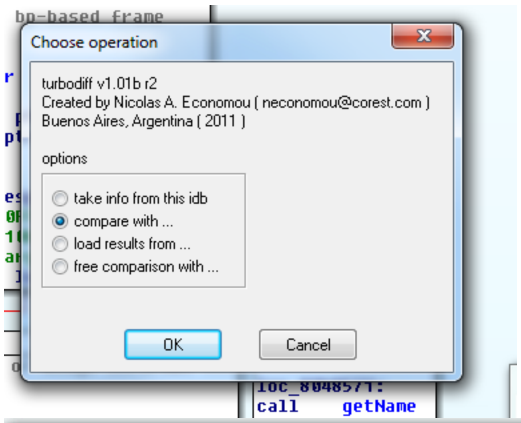
1. You must first start up the free IDA 5.0 application you previously installed. Once it is up and running, go to File | New, select the Unix tab from the popup, and click the ELF option on the left, as shown here, and then click OK.



2. Navigate to your C:\grayhat\app_diff\ folder and select the file "name." Accept the default options that appear. IDA should quickly complete its auto-analysis, defaulting to the **main()** function in the disassembly window.
3. Press CTRL-F11 (or go to the top bar and chose exit\plugin\turbodiff) to bring up the turbodiff pop-up. If it does not appear, go back and ensure you properly copied over the necessary files for turbodiff. With the turbodiff window on the screen, select the option "**take info from this idb**" and click OK, followed by another OK.
4. Go to File | New, and you will get a pop-up box asking if you would like to save the database. Accept the defaults and click OK. Repeat the steps of selecting the UNIX tab | ELF Executable, and then click OK. Open up the name2 ELF binary file and accept the defaults. Repeat the steps of bringing up the turbodiff pop-up and choosing the option "take info from this idb."
5. Now that you have completed this for both files, press CTRL-F11 again, with the name2 file still open in IDA.



6. Select the option "compare with…" and click OK.

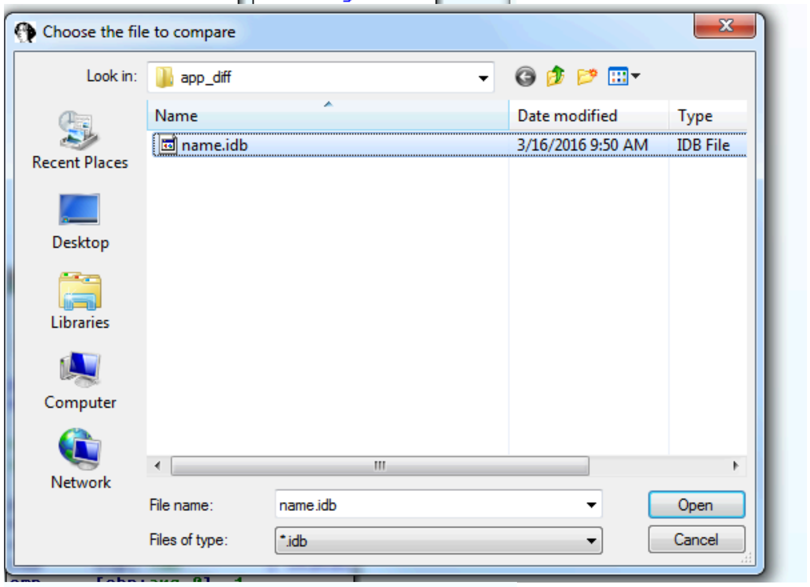


Choose operation

turbodiff v1.01b r2
Created by Nicolas A. Economou (neconomou@corest.com)
Buenos Aires, Argentina (2011)

options

- ○ take info from this idb
- ● compare with ...
- ○ load results from ...
- ○ free comparison with ...

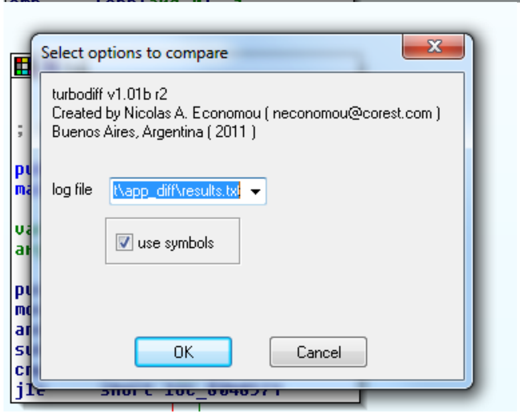


Choose the file to compare

Look in: app_diff

Name	Date modified	Type
name.idb	3/16/2016 9:50 AM	IDB File

File name: name.idb
Files of type: *.idb



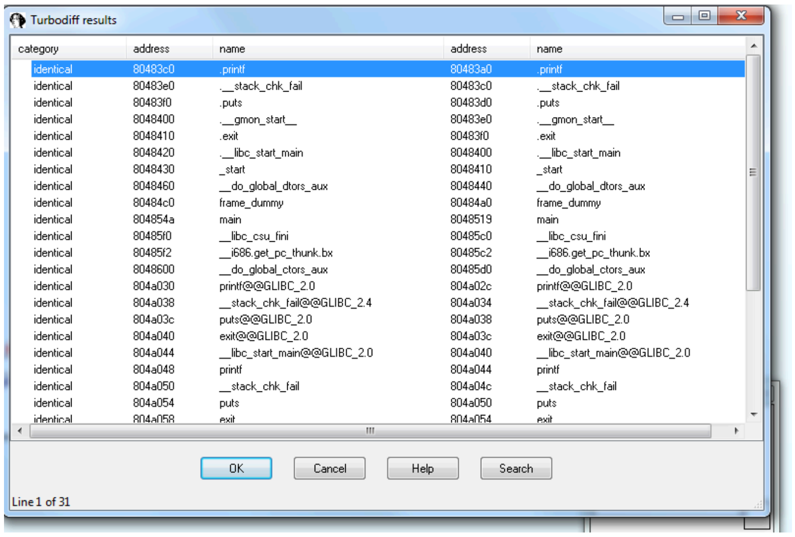
Select options to compare

turbodiff v1.01b r2
Created by Nicolas A. Economou (neconomou@corest.com)
Buenos Aires, Argentina (2011)

log file t:\app_diff\results.txt

☑ use symbols

7. Select the name.idb file and click OK, followed by another OK. The following box should appear (you may have to sort by category to replicate the exact image):

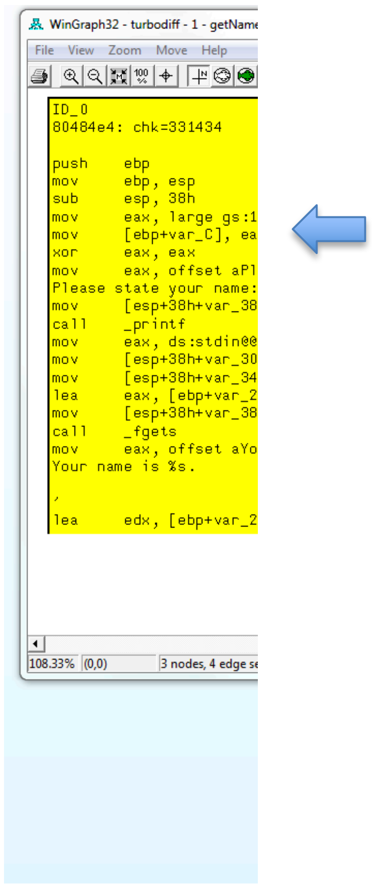


Turbodiff results

category	address	name	address	name
identical	80483c0	.printf	80483a0	.printf
identical	80483e0	.__stack_chk_fail	80483c0	.__stack_chk_fail
identical	80483f0	.puts	80483d0	.puts
identical	8048400	.__gmon_start__	80483e0	.__gmon_start__
identical	8048410	.exit	80483f0	.exit
identical	8048420	.__libc_start_main	8048400	.__libc_start_main
identical	8048430	_start	8048410	_start
identical	8048460	__do_global_dtors_aux	8048440	__do_global_dtors_aux
identical	80484c0	frame_dummy	80484a0	frame_dummy
identical	804854a	main	8048519	main
identical	80485f0	__libc_csu_fini	80485c0	__libc_csu_fini
identical	80485f2	__i686.get_pc_thunk.bx	80485c2	__i686.get_pc_thunk.bx
identical	8048600	__do_global_ctors_aux	80485d0	__do_global_ctors_aux
identical	804a030	printf@@GLIBC_2.0	804a02c	printf@@GLIBC_2.0
identical	804a038	__stack_chk_fail@@GLIBC_2.4	804a034	__stack_chk_fail@@GLIBC_2.4
identical	804a03c	puts@@GLIBC_2.0	804a038	puts@@GLIBC_2.0
identical	804a040	exit@@GLIBC_2.0	804a03c	exit@@GLIBC_2.0
identical	804a044	__libc_start_main@@GLIBC_2.0	804a040	__libc_start_main@@GLIBC_2.0
identical	804a048	printf	804a044	printf
identical	804a050	__stack_chk_fail	804a04c	__stack_chk_fail
identical	804a054	puts	804a050	puts
identical	804a058	exit	804a054	exit

Line 1 of 31

Note that the getName() function is labeled "suspicious ++." Double-click the getName() function to get the following window:



```
ID_0
80484e4: chk=331434

push    ebp
mov     ebp, esp
sub     esp, 38h
mov     eax, large gs:1
mov     [ebp+var_C], ea
xor     eax, eax
mov     eax, offset aPl
Please state your name:
mov     [esp+38h+var_38
call    _printf
mov     eax, ds:stdin@@
mov     [esp+38h+var_30
mov     [esp+38h+var_34
lea     eax, [ebp+var_2
mov     [esp+38h+var_38
call    _fgets
mov     eax, offset aYo
Your name is %s.

,
lea     edx, [ebp+var_2
```

108.33% (0,0) 3 nodes, 4 edge se

In this image, the left window shows the patched function and the right window shows the unpatched function.

- The unpatched block uses the **gets()** function, which provides no bounds checking.
- The patched block uses the **fgets()** function, which requires a size argument to help to prevent buffer overflows. The patched disassembly is shown here:

```
mov eax, ds:stdin@@GLIBC_2_0
mov [esp+38h+var_30], eax
mov [esp+38h+var_34], 14h
lea eax, [ebp+var_20]
mov [esp+38h+var_38], eax
call _fgets
```

There were a couple of additional blocks of code within the two functions, but they are white and include no changed code. They are simply the stack-smashing protector code, which validates stack canaries, followed by the function epilog. At this point, you have completed the lab. Moving forward, we will look at real-world diffs.