Leo Yan -- leoyan
Junli Shao -- junlis

For the framing, we choose “large software development organization.” After comparing two tools, we think that it is better for us to use the Codesonar. As a massive software development organization, it is common to have a large code base. Codesonar is able to run static analysis on such a code base and its security and functionality could really help us with team development. It runs well with IDEs and version control tools such as VS Code and Github. So Codesonar is definitely the better choice for us here.

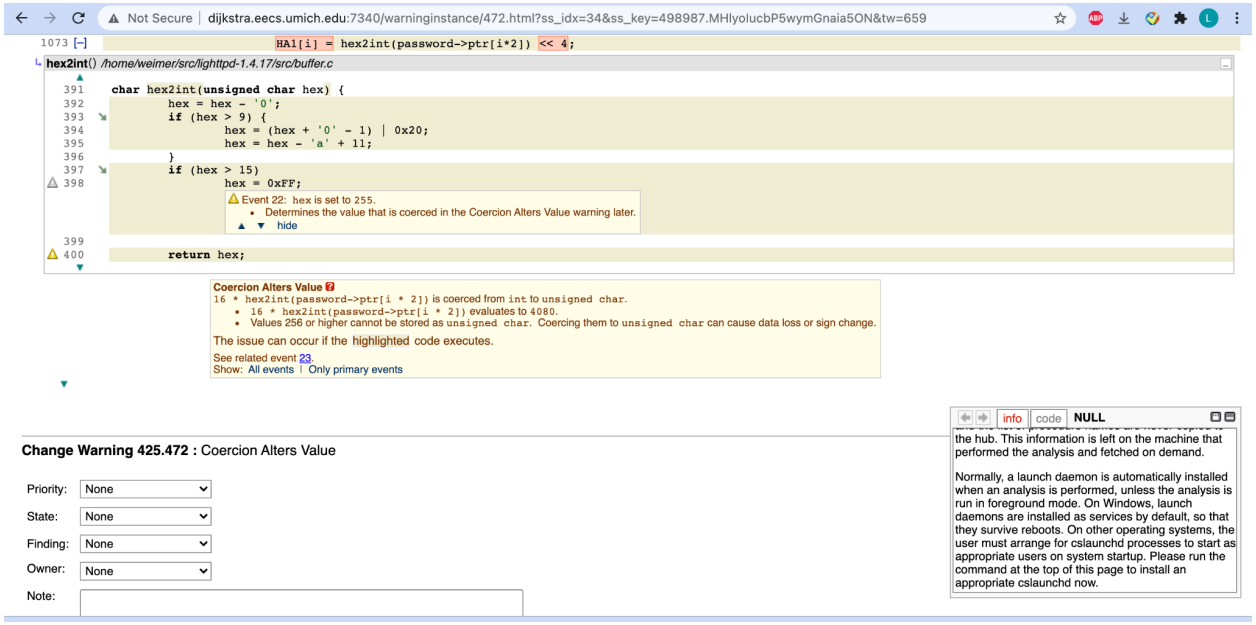
We didn’t not have to do anything for Codesonar since it is pre-run for us. No dependencies or installation is needed in this case. The spec also indicates a way of running, which only takes about 4 commands, so I would assume that it is not very difficult to use the tool. In addition, the reports show that Codesonar runs relatively fast. It took Codesonar about 15 minutes to run on jfreechart and about 18 minutes to run on lighttpd. We would say it is a fast tool even for large code bases. Compared to Codesonar, Infer is more difficult to set up. Some Infer versions are hard to build and install and cannot support some operating systems. For convenience, we select the pre-compiled version of Infer and run it on AWS. After compilation, the Infer is easy to use because only few commands are needed and it is a static analyzer so we don’t have to run the program in order to detect defects. The running time for jfreechart with Infer is about 2 hours and for lighttpd is about 1.5 hours. Therefore, concerning the efficiency of detecting defects, Codesonar wins over Infer.

For codesonar, in order to locate the reports, I have to download and set up UM VPN. After the VPN is connected, I can then log in to the website and see the reports. The reports are very detailed and the website is designed to guide us through the reports and locate the information we want. We can sort the warnings by different categories, including significance, line number and class, which I think are the most helpful categories to locate useful information. For Infer, the report is created inside the Infer folder with the name bugs.txt. The report contains the number of issues found, file name, line numbers of defects, types of error, and a snapshot of part of the defected code. Compared to CodeSonar, Infer generates a less detailed report.

Both tools generated high quality and detailed analysis reports. For Codesonar, as mentioned above, it has many categories to help the developer locate the information. It put the warning into different categorizations, making it easier for the developers. Infer also labels different categorizations of error but errors within the same category are not collected, which is a bit inconvenient. Compared to Infer, I think Codesonar is more detailed, for reasons including simply Codesonar having more categories/classes of warnings. However, there are some duplicate reports within both Codesonar and Infer, for example the report of JFreeChart by Infer contains duplicate issues in DynamicTimeSeriesCollection.java:667, with one states that “Reads without synchronization” and the other states that “Unprotected write”, though the reasons are listed as different, the underlying principle is same because both imply the risk of read something that might be altered at the same time. Moreover, they both reported some overlap of issues since they ran the analysis on the same files and both of them generated warnings and

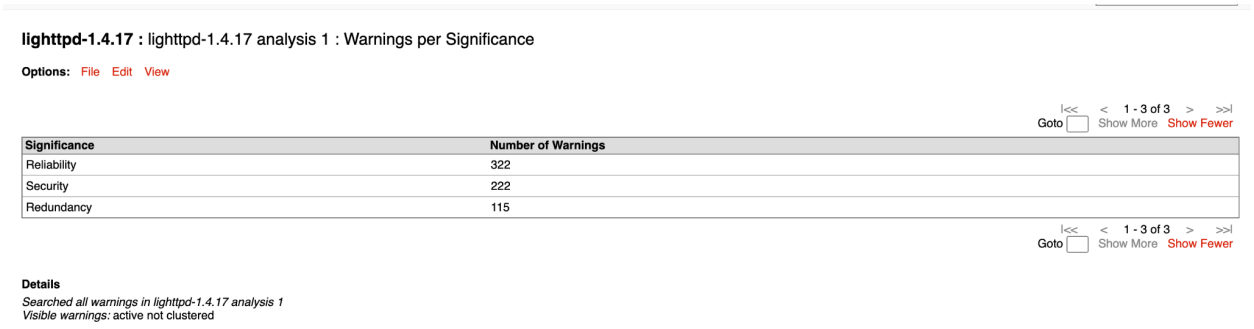
put them into very similar categories. Through our analysis, most of the overlaps are between null pointer dereference and memory leak. Even though Codesonar generally has more detailed reports, there are also some cases that are missing from Codesonar but are reported by Infer such as the file etags.c. There are also costs to use these tools. Obviously both of them take time to run. For Infer, we need to download and install the tool manually, and it may not run very well on some platform/environment so we might need to prepare them for it. As for Codesonar, we need the license to the tool, which usually means it is not free and we have to pay extra money for it.

Firstly we choose CVE-2011-4362, which is known as “Integer signedness error in the base64_decode function in the HTTP authentication functionality (http_auth.c) in lighttpd 1.4 before 1.4.30 and 1.5 before SVN revision 2806 allows remote attackers to cause a denial of service (segmentation fault) via crafted base64 input that triggers an out-of-bounds read with a negative index,” according to the lighttpd CVE website. Based on the Infer analysis report, we don’t think that it successfully found this CVE. Although Infer analysis pointed out some “uninitilized_value” warnings, none of them could really help us to identify CVE-2011-4362. On the other hand, Codesonar successfully pointed out this CVE with a security warning called “Coercion Alters Value” in the http_auth.c file as shown in the screenshot below.



Another CVE identified by the website that we choose is CVE-2014-2323, which is “SQL injection vulnerability in mod_mysql_vhost.c in lighttpd before 1.4.35 allows remote attackers to execute arbitrary SQL commands via the host name, related to request_check_hostname.” Unfortunately, such CVE seems to be missing from both of the analysis tools. We specifically tried to find warnings from both reports inside the mod_mysql_vhost.c file, but neither Codesonar nor Infer pointed out any warnings in the file. Overall, both Codesonar and Infer can find some security defects, but Codesonar seems to be better at detecting such bugs and issues, while Infer could only find some minor ones.

Looking at both reports on the lighttpd program from Codesonar and Infer, we think that Codesonar is the better tool in this case. Although both Codesonar and Infer generated false negative results, which means they both missed some CVEs they should have detected, they did also catch some important ones. For example, they both missed CVE-2014-2323. Beside that, we also found out that Codesonar missed CVE-2008-1531 and CVE-2008-4298. As mentioned above, Codesonar successfully identified CVE-2011-4362. Furthermore, through our analysis, Codesonar has lower rates of getting both false positives and false negatives than Infer when analyzing the lighttpd program. Codesonar is more accurate because it follows an analysis pattern based on paths, the most false positives of Codesonar are about explicit cast but Infer has more false positives related to path issues. Considering an example when the web server free resources in each if statement above but not at the end, Codesonar will analyze each if statement and determine the resources can be free but Infer will only look at the end and determine it is a defection. As for other useful insights, Codesonar found many issues of high priority and severity. Based on the statistical table, it found 322 reliability issues, 222 security issues and 115 redundancy issues. These results gave us the insights of the program that may contain issues with high significance.



lighttpd-1.4.17 : lighttpd-1.4.17 analysis 1 : Warnings per Significance

Options: File Edit View

|<< < 1 - 3 of 3 > >>|
Goto Show More Show Fewer

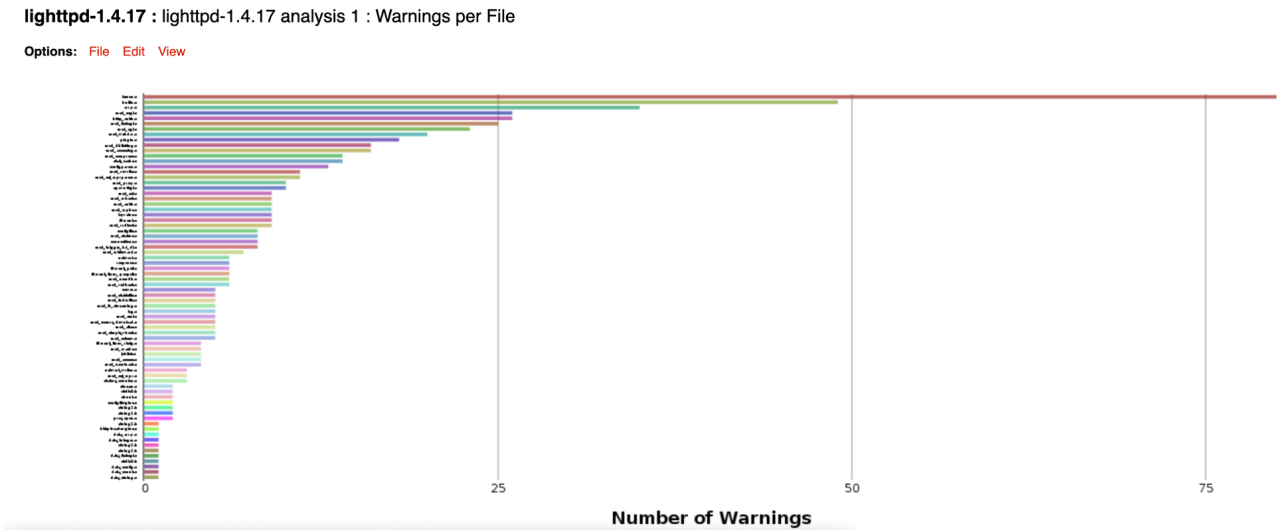
Significance	Number of Warnings
Reliability	322
Security	222
Redundancy	115

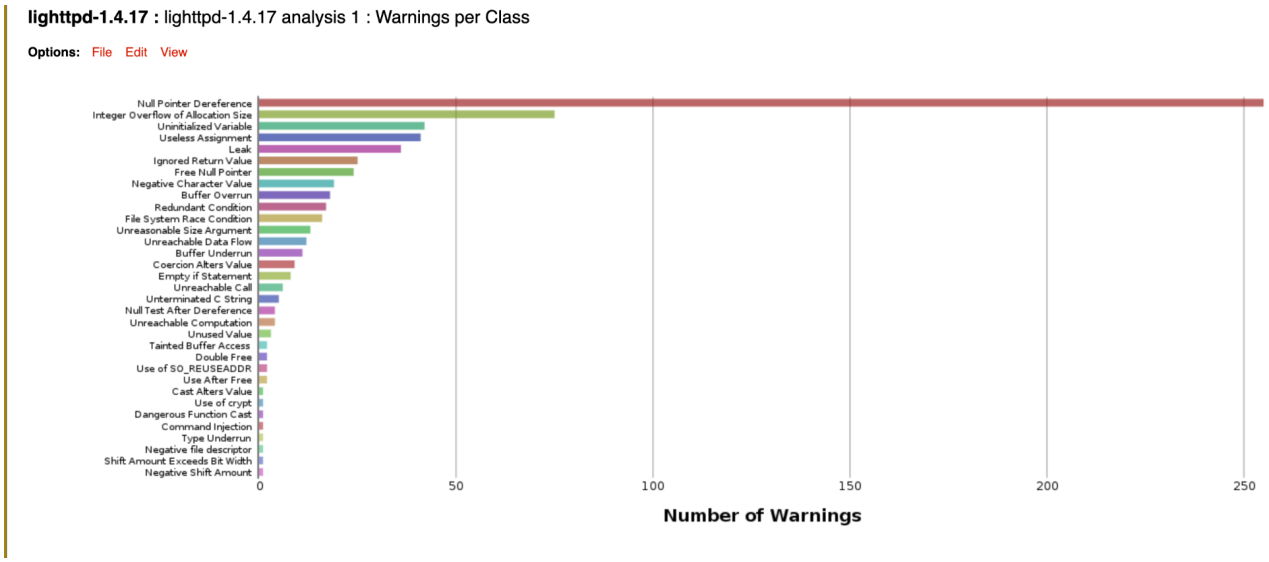
|<< < 1 - 3 of 3 > >>|
Goto Show More Show Fewer

Details

Searched all warnings in lighttpd-1.4.17 analysis 1
Visible warnings: active not clustered

Another two important charts from Codesonar reports are shown below --- number of warnings by class and file. These charts can help the developers to solve the warnings based on some priority they choose. For example, the charts tell us that lemon.c has the most warnings, so it would be reasonable for the developers to start solving these warnings with lemon.c.

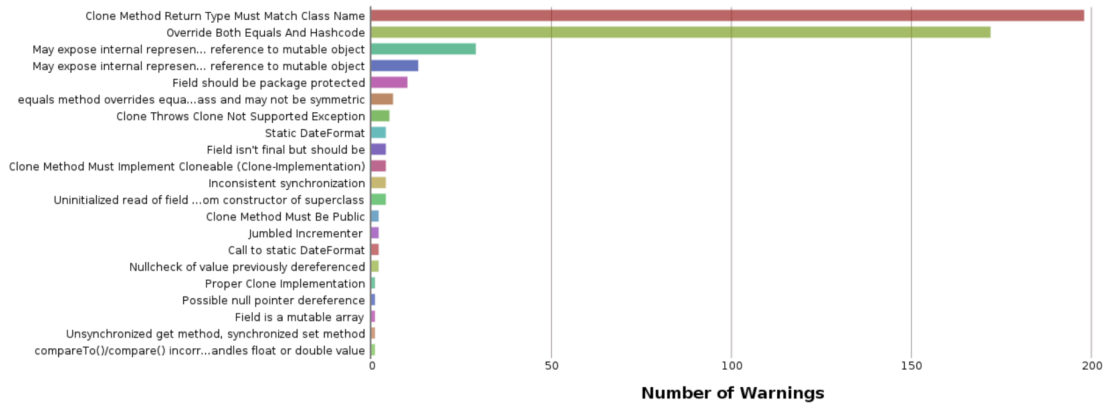




As for the jfreechart program, it seems that Infer is the better tool in this case. From the chart below reported by Codesonar, we can see that it did not give us many warnings that seem to be significant, that is, warnings such as “Clone Method Return Type Must Match Class Name” are not as important as other security or memory leak issues. I think most of the warnings reported by Codesonar are code-written style issues, which should be easy to fix and should not affect performance too much. On the other hand, Infer did give us some useful information on issues within the jfreechart program, which includes resource leak, null dereference, and thread safety violation. For example, one useful warning Infer generates but Codesonar doesn’t is in line DynamicTimeSeriesCollection.java:787, which states the possibility of read/write race because the RegularTimePeriod object tp is reading from a non write-protected data structure of the class “RegularTimePeriod[] pointsInTime”. Through our analysis of the code, though the possibility of race happening is low, it is still possible when DynamicTimeSeriesCollection is used in multi-thread circumstances.

jfreechart-1.5.0 : jfreechart-1.5.0 analysis 1 : Warnings per Class

Options: File Edit View



We choose libpng to be our additional subject program and we think that Codesonar is more helpful here.

|<< < 1 - 20 of 20 > >>|
Goto Show More Show Fewer

Warning Class	Number of Warnings
Useless Assignment	105
Null Pointer Dereference	36
Redundant Condition	17
Uninitialized Variable	10
Unreachable Call	9
Unused Value	8
Unreasonable Size Argument	7
Buffer Overrun	5
File System Race Condition	5
Ignored Return Value	5
Unreachable Computation	4
Tainted Buffer Access	4
Cast Alters Value	3
Buffer Underrun	2
Use of tmpfile	2
Integer Overflow of Allocation Size	2
Float Division By Zero	1
Unreachable Data Flow	1
Unreachable Conditional	1
Empty if Statement	1

|<< < 1 - 20 of 20 > >>|
Goto Show More Show Fewer

Goto Show More Show Fewer

File Name	Number of Warnings
pngvalid.c	116
pngfix.c	19
pngimage.c	16
pngrutil.c	15
png.c	12
pngcp.c	11
pngunknown.c	10
pngstest.c	6
timepng.c	6
pngtest.c	3
pngmem.c	2
pngwrite.c	2
stdio2.h	1
pngwutil.c	1
pngerror.c	1
string3.h	1
string3.h	1
string3.h	1
string3.h	1
string3.h	1
string3.h	1
pngread.c	1

|<< < 1 - 22 of 22 > >>|
Goto Show More Show Fewer

Looking at the two charts gendered by Codesonar, it reported many significant issues such as “Null pointer dereference” and “Float division by zero” along with other safety issues. It also provides us with some insights of some minor issues such as “Useless assignment.”

```
pngrutil.c:3150: error: DEAD_STORE
  The value written to &idat_limit is never used.
  3148.       if (png_ptr->chunk_name == png_IDAT)
  3149.       {
  3150. >        png_alloc_size_t idat_limit = PNG_UINT_31_MAX;
  3151.          size_t row_factor =
  3152.             (png_ptr->width * png_ptr->channels * (png_ptr->bit_depth > 8? 2: 1)

contrib/tools/pngfix.c:3179: error: NULL_DEREFERENCE
  pointer `file` last assigned on line 3177 could be null and is dereferenced at line 3179, column 8.
  3177.       struct file *file = get_control(png_ptr);
  3178.
  3179. >     if (file->global->warnings)
  3180.          emit_error(file, LIBPNG_WARNING_CODE, message);
  3181.
```

However, Infer fails to generate a comprehensive report like Codesonar did, but only points out issues such as Null_Dereference and Dead_Store. Above is a screenshot of some warnings reported by Infer. Most of the others look like this. The warnings reported by Infer are very limited, so it must have many false negatives. Therefore, based on the quality of the reports from both tools, Codesonar is the better one for the libpng program.

In conclusion, we could recommend Codesonar for our supervisor. The first reason is that Codesonar has better usability and readability. As a large software organization, we want to provide our developers with tools that they could feel ease to use. Codesonar gives results on the website with a clean interface. The developers can easily sort the warnings based on different categories and quickly locate the information they want. The charts and tables generated by the Codesonar are also very helpful for the developers when analyzing the potential issues. Secondly, Codesonar generates more detailed reports in most cases. They have less false positives and false negatives in general compared to Infer. It is important for the developers to use a tool that is comprehensive enough. At least the tool can help them to find the major issues. Lastly, Codesonar can be integrated with IDEs and version control tools, which is extremely important for large software development organizations and team based projects.

Even though we may need to pay for the Codesonar licenses, it is totally worth it. In addition, Codesonar seems to be more efficient in runtime compared to infer.