Obligatory exercise 4: Vulnerabilities in SecureDesktop

Written by, Ole Eivind Parken, StudentID: 121086

October 8, 2014

1 Introduction

In this given assignment we will accomplish these given goals based the source code for a remote desktop program handed out:

- *What is your strategy for the review?*
- *How do you prioritise vulnerabilities found?*
- *How would you modify the application to remove the vulnerabilities?*
- *Document your process so that your results can be reproduced.*

2 Beginning the Review

First I find the files handed out find out that this is files mostly with *.pas* extension. That means that the files are pascal/delphi files. After browsing after a analyzer tool, I downloaded a evaluation version of *Pascal Analyzer*[1] and *CodeHealer*[2]. Later on I needed to enter the files manually to find the errors from the analysis.

2.1 Pascal Analyzer

Pascal Analyzer (PAL) [2] , parses the source code and gives out large tables of information about your code. Unfortunately the evaluation version of Pascal Analyser does not give me all the information about all warnings found (See Figure 1). By clicking on the errors in the report I get to see the line where warnings where found(See Figure 2).

2.2 CodeHealer

At first I didnt understand how to use it when I had problems to start a new project using the *.dpr* files in the directory. I then added the files by inserting them as source files in project preferences. When I then tried to analyze them in CodeHealer [2], I got errors about missing semicolons. I clicked no to edit the source files. Again gives me a new error about that it cant find the required files needed, before it finally does something. After using much time to understand if CodeHealer really does give me some useful information, i decided to use more time on Pascal Analyser.

3 Review & prioritise vulnerabilities

By looking at the lines that Pascal Analyzer found and by reviewing the code in a text editor, I found that there is much hardcoding of paths. This some thing we want to exclude as much as we can, since an attacker can manipulate the files in the hardcoded paths. A good example is *SDSecureDisplaySvoUnit.pas* lines: 368, 491. And there is more but these are commented out, I'm not including those.

It is much code here that should be removed when its commented out and never used. And its almost none comments on what functions does, or why global variables are defined. Which makes the code hard to understand

After doing this, then you can start focus on vulnerabilities in buffer overflow and usage of url addresses to fetch information

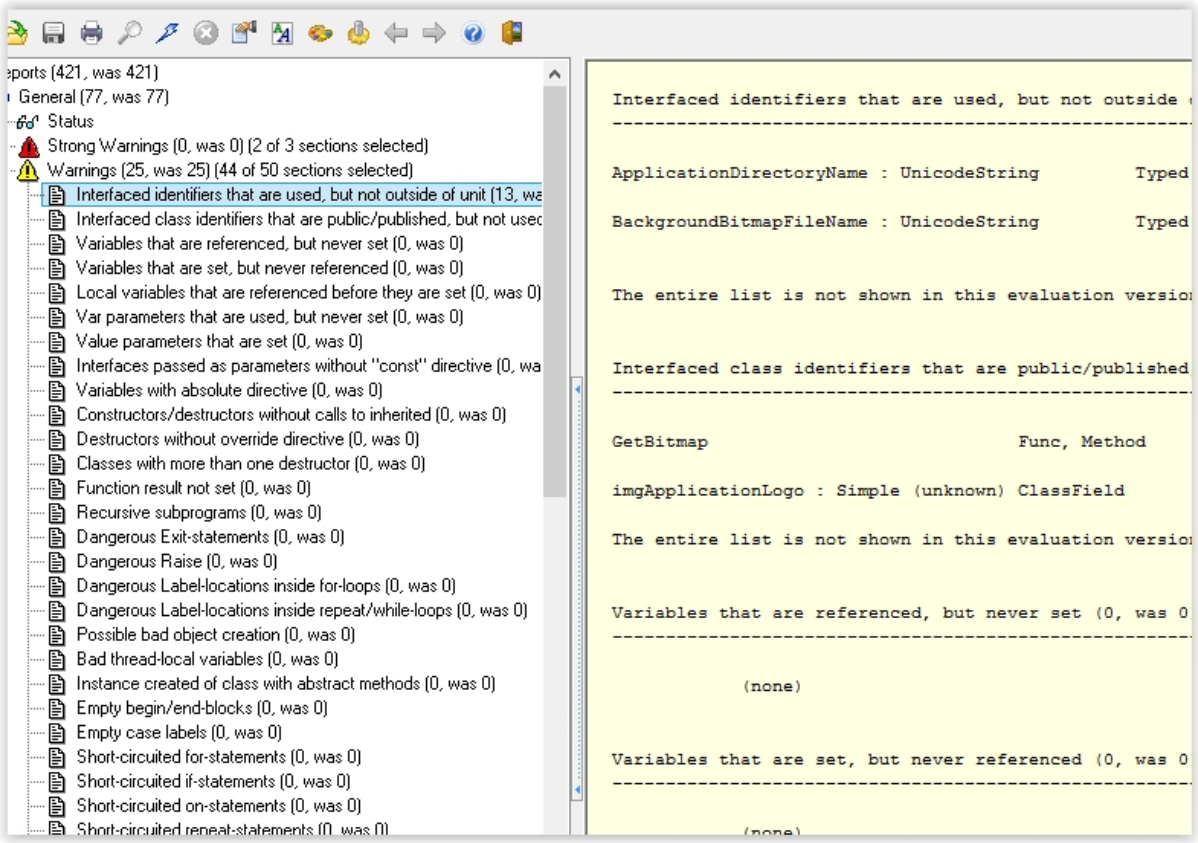

Figure 1: Screen shot from the Pascal Analyzer report

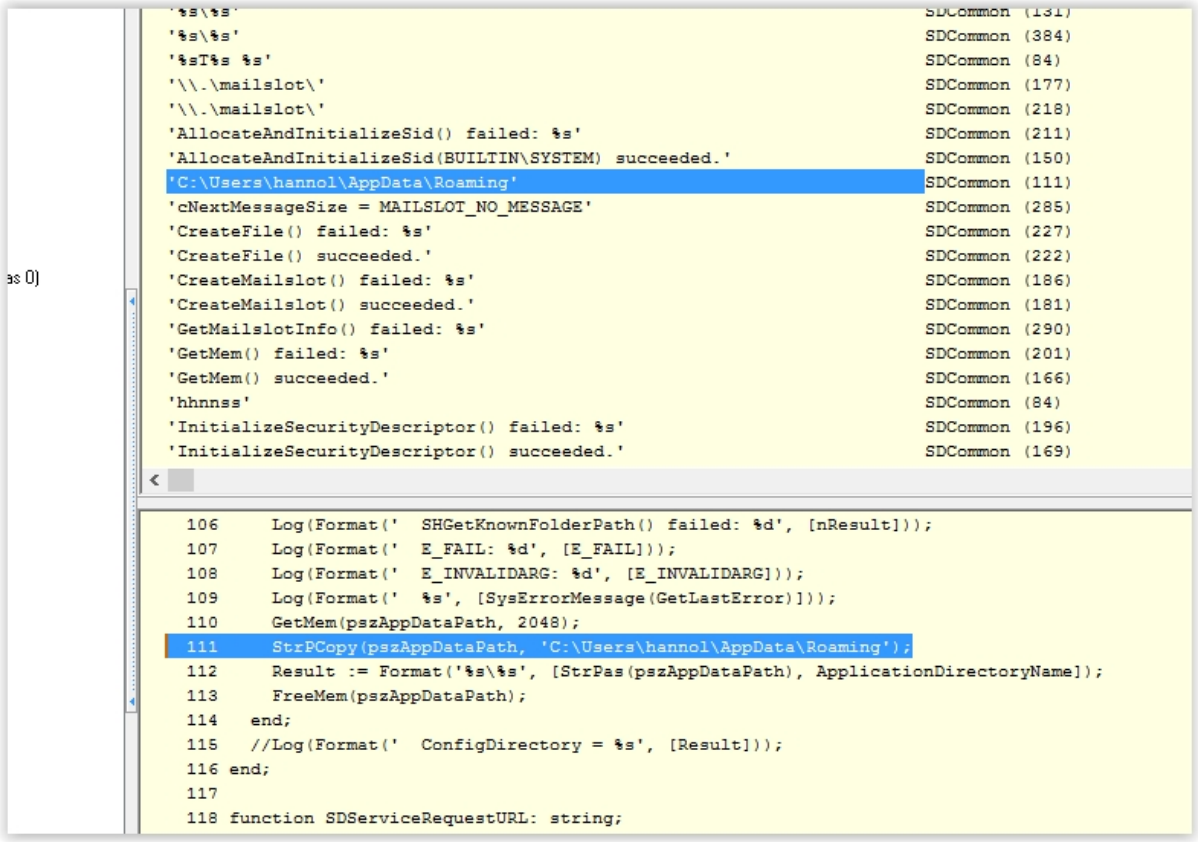

Figure 2: Screen shot from the Pascal Analyzer report, with code review

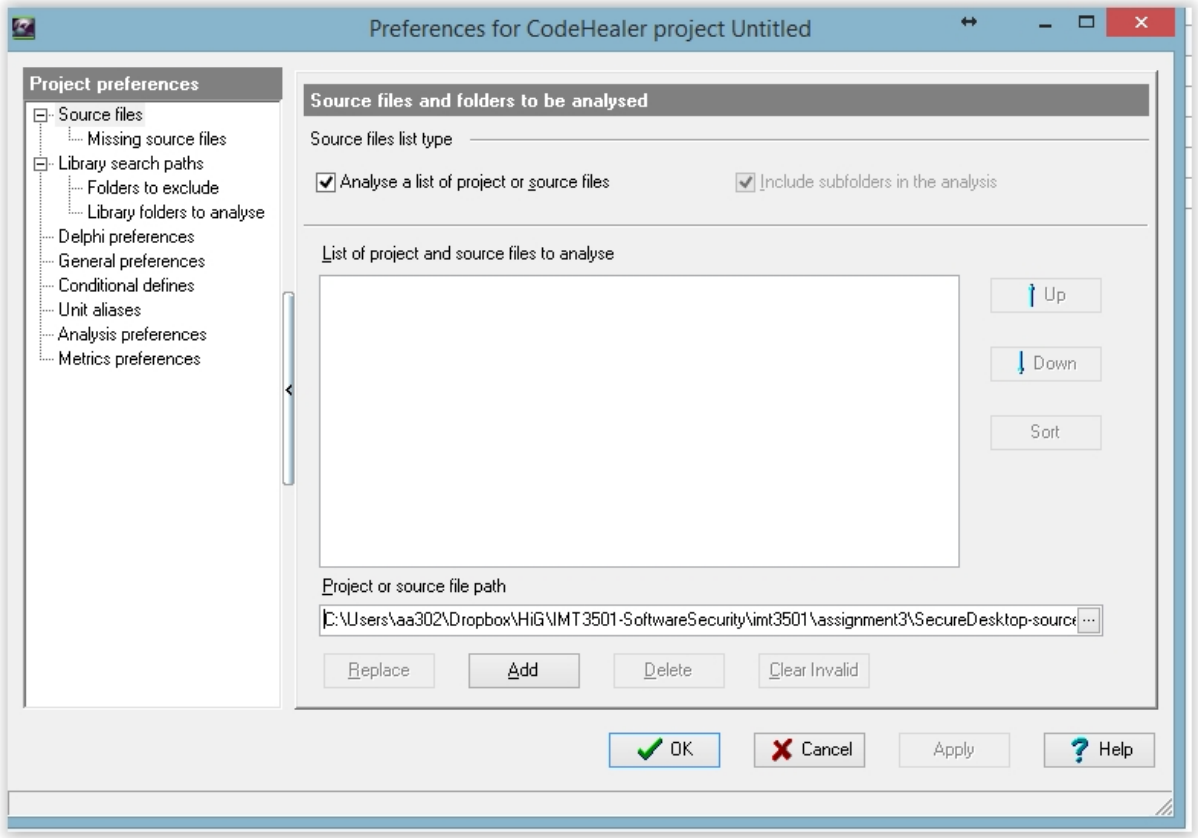

Figure 3: Adding the source files to CodeHealer

References

[1] Peganza, "Pascal Analyzer," http://www.peganza.com/, [Online; accessed 08.10.2014].

[2] SockSoftware, "CodeHealerGroup," http://www.socksoftware.com/default.php, [Online; accessed 06.10.2014].