Mandatory assignment 3
Software vulnerability

- AV: Jan Petter Berg Nilsen
- Studnr: 120505

Intro

This is the there'd mandatory assignment in IMT3501 Software security. In this assignment we are going to look at some pascal source code to find possible vulnerability, that potentially could be exploited by an attacker.

Secure Desktop applications source code revolves around the article Shark Cage. I read the article before I started to do the analyse of the source code, so I was sure of that I understanding what the program was for and it helped me understand the code a little bit more even though I have never seen or programmed in pascal before.

The pascal source code did not have much commenting on the code, so that made it harder to understand the code and to fine vulnerability for the applications. From this I could say that this is a potential vulnerability, because it would mean that the security scan of the code would not be optimal.

When doing the security scan for this application I tryd to use a variety of tools to scan the code, before I vent true all the files in the repository manually to see if I found any vulnerability that way. The different tools that I used and possible vulnerability I have found is listed below.

Tools

The different tools I used to analyse the code found mainly errors which I don't see as a poetensiel threat or vulnerability, but only errors that can be corrected to optimalize the coden and the application. The reason for I didn't find any big errors with the tools could be because I only used the free versions and it would most likely show more bigger errors if I had the paid edition. When I locked up the errors after the analyse I also only got to see 2 or 1 result depending on

the tool. Here is some of the tools that I tried to gather information about the source code:

Eyebol Is a analysing tool that helps find errors and help apply optimizations to in the code, but also generate other information about the source code. sadly this tool gave me not so much result because I used the free version and didn't help me so much in the assignment.

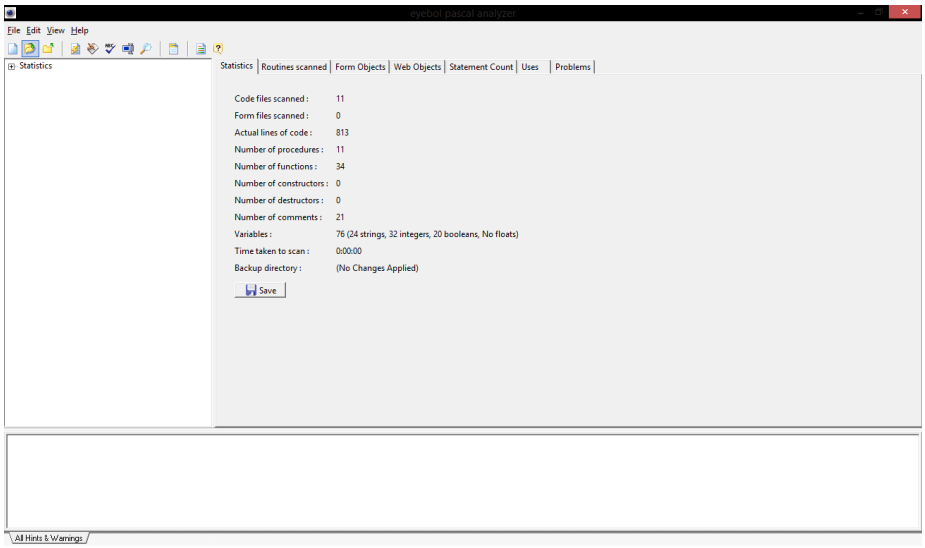

Figure 1: Eyebol

Pascal analyse 7 Is a tool like Eyebol, but so much bather even if I used the free version from both of them. Pascal analyse spots many types of programming bugs and anomalies. In my case it only gave me miner errors. Pascal analyse also generate a lot of information about the source code. Among the analyse tool this was the best one and gave most results and was very easy to use.

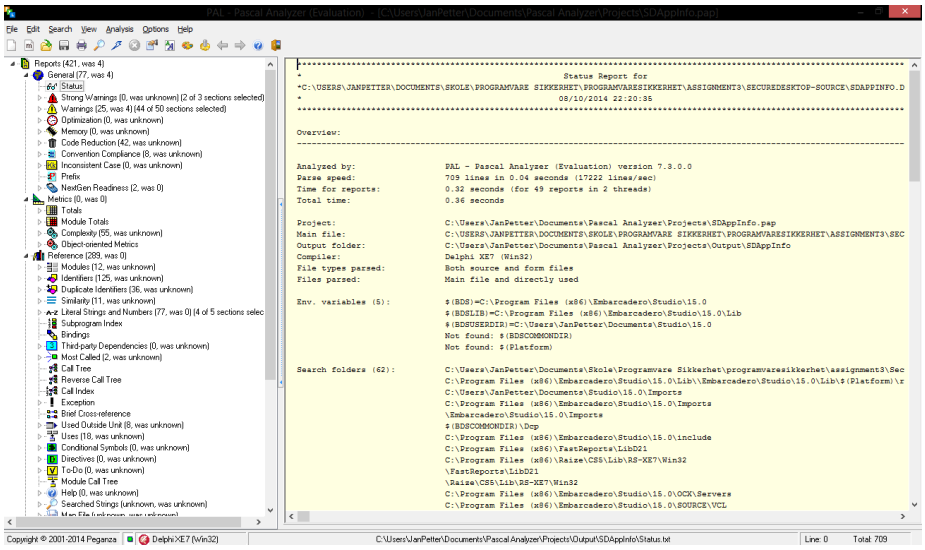

Figure 2: Pascal analyse 7

When doing the manual searching process for vulnerability I used notepad++.

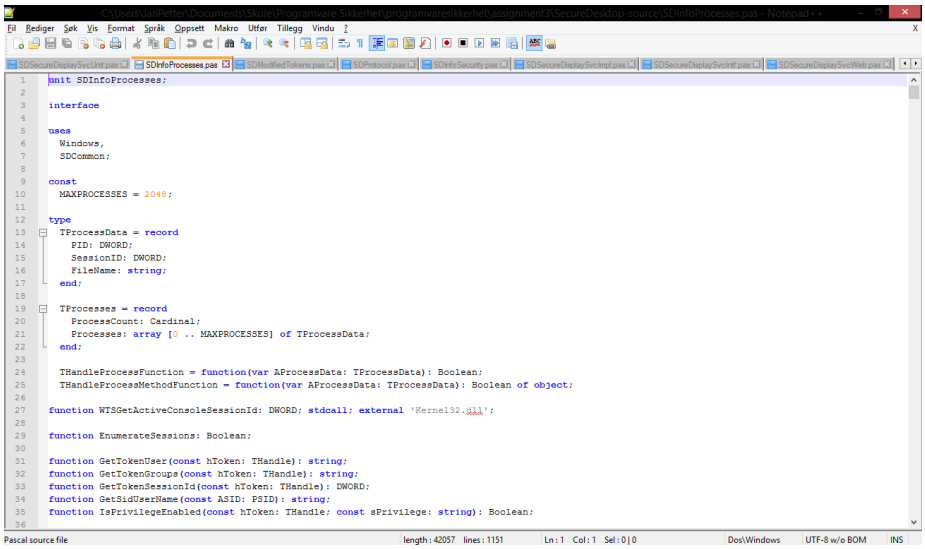

Figure 3: notepad++

I also tried other types of tools like Pascal Browser 2 and Icarus. They gave some information about the source code, but I feel that the other tools mentioned above, helped me more to analyse the different source codes. Other tools that I tried but gave me error, and did not work with the source codes, was Codehealer and DUnit Xtreme Unit Testing for Delphi. For the Delphi I got the error that it was missing files from the code, so this could mean that we didn't get all the source files for the application.

Vulnerability

I didn't find any big vulnerability for either doing the automatic analysing tools or the manual analysis. The manual analysing process was very hard, because it was many lines of code and was poorly commentated. As mentioned in the intro, I would say that bad commenting on the code is no good practise and would make a vulnerability analyse much more difficult, so that it would be easier to over see some potential vulnerability. That way the analyse of the source code would not be as optimized as it could be. When the code is poorly commentated it would also mean that it is harder to maintain, and potential vulnerability could occur.

Here is some examples of errors that I found, when doing the automated search with the different tools:

- Redeclared identifiers from System unit in the files fmSDAppInfo line 199.
- Variables that are set, but never referenced in file SDInfoProcesses line 625 and 626.

- Interfaced identifiers that are used, but not outside of unit in file SDCommon line 15 and 23.
- Redeclared identifiers from System unit in SDUserSessionManager line 34.

Most of these error could easily be fixed and I really do not look at them as a type of vulnerability, but they could also occur because of some missing files that I thought about later in the rapport.

When I did the manual search I only found that the program used a lot of hard coded values and paths, like it does on the file SDcommand.pas on line 111. This is something that could be fixed by using const in the program. I also see that some parts of the source code, like SDInfoSecurity, execute exe and other file types that is not a part of the original program. This could mean that someone could manipulate the program an run malicious software that could potentially be a big vulnerability. What we can do to decrease the probability that an attack like this will occur, is to make some sort of a checking code or authentication, like a sort of code that is implemented to check the exe file before it starts running. This will only make sure that the right executable files that can execute the program.

Bibliography

[1] analyse tools, for analyse tool http://www.torry.net/pages.php?id=50

[2] analyse tools, http://delphi.about.com/od/devutilities/ss/icarus-delphi-unit-uses-clause-analyzer.htm