DV1667

Anti Virus Program Requirements 2

Viktor Listi
vili22@student.bth.se

ID	Name	Description	Testing	Dependencies	TBI (YES/NO)
A.1	Function(FileRecursive)	Enables the code to iterate through directory hierarchy and list all the files contained in a vector.	Manual testing with different operating systems and different folder hierarchies.	G.1	YES
B.1	Function(VirusDatabaseIntegrityChecker)	Verifies that the virus database is the right structure, virus name followed by “=” followed by the virus description.	Manual testing with databases that contain known errors.	G.1	YES
C.1	Function(VirusNames)	Extracts the virus names from the database and lists all of them in a vector.	Manual testing by comparing the output vector to the real database.	G.1, B.1	YES
D.1	Function(VirusDescriptions)	Extracts the virus descriptions in hexadecimal form from the database and lists all of them in a vector.	Manual testing by comparing the output vector to the real database.	G.1, B.1	YES
E.1	Function(ConvertHex)	Converts the hexadecimal virus descriptions to ASCII characters and saves the converted descriptions in a vector.	Manual testing by manually converting the hexadecimal of the database and comparing to the output vector.	D.1	YES
F.1	Function(FlagFiles)	Compares and flags all the files in the folder hierarchy that match one of the virus descriptions.	Manual testing by creating a scenario where the virus files are known and seeing if it flags them correctly.	A.1, E.1, C.1	YES
G.1	User Input	Allows the user to input a folder hierarchy and virus database.	Manual testing.	N/A	YES
G.2	User input error handling	Checks the user input if it matches set allowed file extensions, entry lengths and if the files exist.	Manual testing using non-allowed file extensions and non-existing files.	G.1	YES
H.1	Output to log file	Writes the contents of the vector that contains flagged files to an external text document.	Manual testing by examining the resulting text file to the FlagFiles vector.	F.1	YES
I.1	Too many entries error handling	Checks for the amount of entries in the names, files and descriptions vectors in order to stop heap overflow and slow run time.	Manual testing by running the program with databases that are too large.	A.1, C.1, D.1	YES

J.1	Syntax error handling	Checks for syntax errors in the virus database that would stop the program from running.	Manual testing by creating scenarios with known syntax errors and checking if it catches them.	G.1, B.1	YES

Working with requirements

I worked with these requirements by setting up a plan of what functions and features the anti-virus program needed to have and what each of the requirements would depend on for the program to work. This was useful as to get a full picture of the program before initiating the coding process. During part 2 of the assignment these requirements were used and compared to the vulnerability report in order to see which problems could occur and needed to be fixed. Since the last version of the program 6 different improvements have been made based on the vulnerability analysis. These are the following: An improved directory iteration function that can handle Windows OS as well as Linux OS, heap overflow protection, an improved description comparator, reinforced syntax error checking, improved user input validation, and the inclusion of const in the function parameters in order to make it more secure.

The ability for the software to be able to handle Windows OS in addition to Linux OS is because with the previous implementation of the software it could only handle Linux based operating systems, so the ability to handle Windows based systems is an important improvement as it should be able to handle all possibilities.

A method for heap overflow protection was implemented as it was one of the vulnerabilities identified in the vulnerability report with the highest score as it could easily cause the softwares integrity to fail.

The improved description comparator was an important improvement from the previous implementation of the software. It allows the software to detect more potential viruses than before which increases the reliability of it detecting all the viruses in the folder hierarchy by a lot.

A method for reinforcing the syntax error checking was an important improvement as previously it would not detect any syntax error which could prompt false positives in the final results or make it completely miss potential viruses.

The improved user input validation allows the program to check the input of the user before it can even come to the point of failure which is a very important improvement. It also stops potential foul actors from using the input part of the software to create scenarios where the source code vulnerabilities could be taken advantage of.

The reason for addressing just these vulnerabilities in the code is due to them being of high threat to the integrity of the software. With these vulnerabilities fixed the program should be robust to the point of where it is very difficult for it to fail, if possible at all. Before these changes were made the software could easily fail when encountering problems that it could very easily face in regular use such as wrong file paths or the files not existing at all. The current version could easily prevent this from being a problem as well as even preventing targeted attacks against it. The return on investment from implementing these easy fail safes is very high as the security they have added compared to the time it took to implement them and the unnoticeably longer run time is very low. With these changes included the software has ridden itself of most of its downfalls and is in the end a more secure program.