Static Analysis:

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

EncryptFileA is not a typical function import that could be malicious, this leads me to think that this malware sample could potentially be encrypting files on the system.

MessageBoxW isn’t malicious in itself, but this is the first time we have seen it in our samples and the flags that can be used with it could make the message box represent malicious intent.

There was also a IsDebuggerPresent import which could also represent something malicious because the program may not want to run if a debugger is present.

OllyDbg:

Text

Description automatically generated

The program’s first call is GetConsoleWindow which then returns the handle for the console window and has no parameters. This handle is then used in the ShowWindow call which has arguments for the handle to the window and how the window is shown, the parameter contained hides the window so that the console is no longer visible to the user.

Next the program calls GetLocalTime, and the return value is the time. This is a step that I had to pay extra attention to because after the time was returned for the system, it was then compared to the hex value 7EE which the decimal value is 2030. The local time returned for my system was decimal 2023, and since it did not match it resulted in the program taking a pathway that resulted in the program looping around and not doing anything further. This was preventing the program from successfully running until the local time for the year matched.

By changing the value of the time before stepping further into the program, it would then match and move on to the next call.

A picture containing text

Description automatically generated

A call to GetEnvironmentVariableA is made for the USERPROFILE variable, a pointer to 001BFA04, and a buffer size of 80. When this is executed the environment variable for USERPROFILE is found and stored as shown below.

Table

Description automatically generated

The next event is when the call to strcat is made a few different times. This call is taking the src parameter and then appending that onto the dest parameter. The first call takes the “C:\Users\sysadmin” dest parameter that the GetEnvironmentVariableA had returned and appends “\Doc” onto it. So that in the eax register the value in ASCII is “C:\Users\sysadmin\Doc”.

Text

Description automatically generated

The next strcat call appends “u” onto the eax register contents so that it becomes “C:\Users\sysadmin\Docu” in the eax register.

Text

Description automatically generated

The next strcat call appends “men” onto the eax register contents so that it becomes “C:\Users\sysadmin\Documen” in the eax register.

Text

Description automatically generated

The next strcat call appends “ts\” onto the eax register contents so that it becomes “C:\Users\sysadmin\Documents\” onto the eax register, so now there is a clear path that the program wants to access the documents folder on the system.

Text

Description automatically generated

The next call is a \_chdir command in the hidden console window. The only argument used with this call is the path that was just created and is contained in the eax register. This changes the directory of the cmd window to the Documents folder on the system of the USERPROFILE variable, in this case my user is sysadmin.

Table

Description automatically generated

Next the program jumps to a function where there is a search for any files contained in this folder. A call to FindFirstFileA is made where the filename parameter is “\*.docx” and the pointer to where the found file goes is 001BF798.

For this example, I had a few .docx files in the Documents folder and after this call this document was found called “passwords.docx”.

Graphical user interface, text, application

Description automatically generatedA picture containing text

Description automatically generated

If a file is found, then there is a jump to a function where EncryptFileA is called and takes the parameter of the file name. The file name is pushed from the eax register after a lea function then the call to EncryptFileA is made, thus then the file is encrypted. Then a call to FindNextFileA is used to see if there is another file in the folder using the search handle returned from FindFirstFileA and the pointer to where to store the file name. If the return value succeeds and the value is a nonzero, so another file is found, a jump back to the beginning of the encryption loop takes place. Then the file is encrypted, another call to FindNextFileA is made, and if successful repeats again until there are no more files.

A screenshot of a computer

Description automatically generated with medium confidence

After all of the files are encrypted, a call to FindClose is made using the search handle and a return value of nonzero will be returned if successful. Then the function returns back to the main function.

The next call that is made by the program is a call to MessageBox with a few specific parameters. The handle to the owner window is null so there is no owner window. The text is “CRITICAL ALERT! Your files have been Encrypted !!!”. The title of the box is “CRITICAL ALERT” and the other parameters add a stop-sign icon, and ok button on the text box, and a parameter that makes the user respond to the message box before they can continue working in that window.

Graphical user interface, text

Description automatically generated

Graphical user interface, text

Description automatically generated

The next call is GetModuleHandleW which has a null parameter and that means it will return a handle to the file that was used to create the calling process.

Graphical user interface, text, application

Description automatically generated

Finally, the program jumps to a call command of exit with a parameter of status 0, this means that the program was successful and sends the exit command to close the command window.



At this point the program is terminated and the files have been encrypted by the program.

Graphical user interface, application, table

Description automatically generated

Conclusion:

The given sample will first use the call ShowWindow to hide the console window from the user after the program has been executed. After that, there is a call to GetLocalTime where the value of interest returned is the current year of the system. After that call returns a value, that value is compared to the hex value 07EE, which in decimal form is 2030. If the value does not match, then the program takes a different jump and does not continue. If the value matches, then the program moves on to calling GetEnviromentVariableA to get the variable USERPROFILE for the system. That is stored in the buffer pointer parameter specified and is then loaded into the eax register. A few calls to strcat are then made so that the USERPROFILE variable can be appended. The first strcat call adds “\Doc”, the second strcat “u”, the third strcat “men”, and the fourth strcat “ts\”. This results in the value of “C:\Users\sysadmin\Documents\” being stored in the eax register where in this case sysadmin is the value of USERPROFILE that was found using the GetEnvironmentVariableA call. The program then calls \_chdir using the file path that was just constructed so that the cmd window moves into the Documents folder. Now the program jumps to a function where a call to FindFirstFileA is made in the Documents folder looking for a document with “\*.docx” so it is looking for a file of any name with that extension. If successful it returns the file name and that is stored for further use. If a file is successfully found, then the program enters an encryption loop. A call to EncryptFileA is made using the file name that was returned, so the file is encrypted, and then a call to FindNextFileA is made to see if there is another file. If there is, a nonzero value will be returned and then a jump to the beginning of the encryption loop is made where the name of the file found is used for the EncryptFileA call. This repeats until no more files are found and the FindNextFileA call returns a zero value. Before returning, a call to FindClose is made using the file search handle. The program will then return to the main function and proceed. The next step taken is that a call to MessageBox is made with multiple parameters consisting of an ok button, the title “CRITICAL ALERT”, the text “"CRITICAL ALERT! Your files have been Encrypted !!!", a stop-sign icon, and the box needs to be closed before continuing to work in the given window. This type of message box would certainly surprise and concern an individual if they unknowingly ran this sample. Next a call to GetModuleHandle is made with a null parameter so that the returned value is a handle to the file used to create the calling process .exe. The program will then jump to the final call where there is a call to exit using status 0 which indicates that it was a success and the cmd window will then exit. At this point the sample program will have been terminated. It seems that the goal of this sample is to find the path to the Documents folder on the system, encrypt any .docx files, notify the user that their files were encrypted using a message box, and then terminate. The program will not run unless the GetLocalTime call returns the value 07EE.