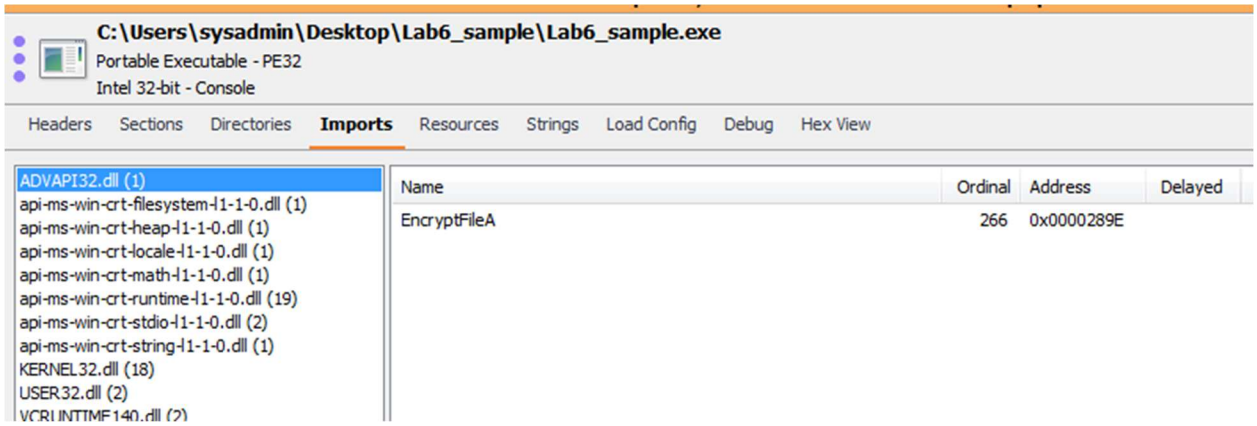
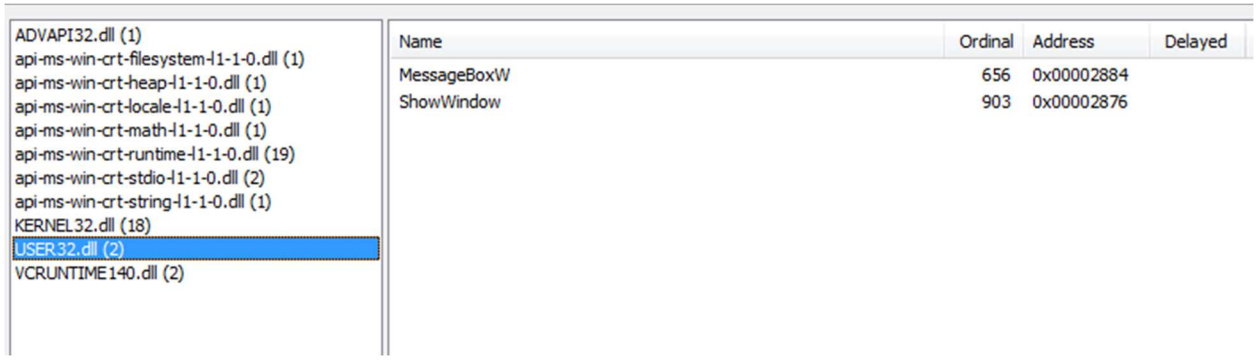
Lab 6
Gunnar Yonker

Static Analysis:





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.

Lab 6
Gunnar Yonker

OllyDbg:

```
00881067  $ 55              PUSH    EBP                                      main
00881068  . 8BEC            MOV     EBP, ESP
0088106A  . 81EC 94000000   SUB     ESP, 94
00881070  . A1 04308800     MOV     EAX, DWORD PTR DS:[883004]
00881075  . 33C5            XOR     EAX, EBP
00881077  . 8945 FC         MOV     DWORD PTR SS:[EBP-4], EAX
0088107A  . 6A 00           PUSH    0
0088107C  . FF15 24208800   CALL    DWORD PTR DS:[<&KERNEL32.GetCons  kernel32.GetConsoleWindow
00881082  . 50              PUSH    EAX                               hWnd
00881083  . FF15 58208800   CALL    DWORD PTR DS:[<&USER32.ShowWind   ShowWindow
00881089  . 8D85 6CFFFFFF   LEA     EAX, DWORD PTR SS:[EBP-94]
0088108F  . 50              PUSH    EAX                               pLocaltime
00881090  . FF15 10208800   CALL    DWORD PTR DS:[<&KERNEL32.GetLoca  GetLocalTime
00881096  . B8 EE070000     MOV     EAX, 7EE
0088109B  . 66:3985 6CFFF   CMP     WORD PTR SS:[EBP-94], AX
008810A2  .v 0F85 88000000  JNZ     Lab6_sam.00881130
```

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.

```
00881096  . B8 EE070000     MOV     EAX, 7EE
0088109B  . 66:3985 6CFFF   CMP     WORD PTR SS:[EBP-94], AX
008810A2  .v 0F85 88000000  JNZ     Lab6_sam.00881130
008810A8  . 68 80000000     PUSH    80                                BufSize = 80 (128.)
008810AD  . 8D85 7CFFFFFF   LEA     EAX, DWORD PTR SS:[EBP-84]
008810B3  . 50              PUSH    EAX                               Buffer
008810B4  . 68 28218800     PUSH    Lab6_sam.00882128                 VarName = "USERPROFILE"
008810B9  . FF15 28208800   CALL    DWORD PTR DS:[<&KERNEL32.GetEnv   GetEnvironmentVariableA
008810BF  . 8D85 7CFFFFFF   LEA     EAX, DWORD PTR SS:[EBP-84]
008810C5  . 68 34218800     PUSH    Lab6_sam.00882134                 src = "\Doc"
008810CA  . 50              PUSH    EAX                               dest
008810CB  . E8 CD0B0000     CALL    <JMP.&api-ms-win-crt-string-11-   strcat
008810D0  . 8D85 7CFFFFFF   LEA     EAX, DWORD PTR SS:[EBP-84]
008810D6  . 68 3C218800     PUSH    Lab6_sam.0088213C                 src = "u"
008810DB  . 50              PUSH    EAX                               dest
008810DC  . E8 BC0B0000     CALL    <JMP.&api-ms-win-crt-string-11-   strcat
008810E1  . 8D85 7CFFFFFF   LEA     EAX, DWORD PTR SS:[EBP-84]
008810E7  . 68 40218800     PUSH    Lab6_sam.00882140                 src = "men"
008810EC  . 50              PUSH    EAX                               dest
008810ED  . E8 AB0B0000     CALL    <JMP.&api-ms-win-crt-string-11-   strcat
008810F2  . 8D85 7CFFFFFF   LEA     EAX, DWORD PTR SS:[EBP-84]
AX=07EE
Stack SS:[001BF9F4]=07EE
```

Lab 6
Gunnar Yonker

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.

```
Stack address=001BFA04, (ASCII "C:\Users\sysadmin")
EAX=00000011

Address   Hex dump                 ASCII
001BFA04  43 3A 5C 55 73 65 72 73  C:\Users
001BFA0C  5C 73 79 73 61 64 6D 69  \sysadmi
001BFA14  6E 00 00 00 40 FA 1B 00  n...@ú←.
```

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".

```
008810C5  . 68 34218800   PUSH   Lab6_sam.00882134                 src = "\Doc"
008810CA  . 50            PUSH   EAX                               dest
008810CB  . E8 CD0B0000   CALL   <JMP.&api-ms-win-crt-string-l1-    strcat
008810D0  . 8D85 7CFFFFFF LEA    EAX, DWORD PTR SS:[EBP-84]
Stack address=001BFA04, (ASCII "C:\Users\sysadmin\Doc")
EAX=001BFA04, (ASCII "C:\Users\sysadmin\Doc")
```

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

```
008810D6  . 68 3C218800   PUSH   Lab6_sam.0088213C                 src = "u"
008810DB  . 50            PUSH   EAX                               dest
008810DC  . E8 BC0B0000   CALL   <JMP.&api-ms-win-crt-string-l1-    strcat
008810E1  . 8D85 7CFFFFFF LEA    EAX, DWORD PTR SS:[EBP-84]
Stack address=001BFA04, (ASCII "C:\Users\sysadmin\Docu")
EAX=001BFA04, (ASCII "C:\Users\sysadmin\Docu")
```

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

```
008810E7  . 68 40218800   PUSH   Lab6_sam.00882140                 src = "men"
008810EC  . 50            PUSH   EAX                               dest
008810ED  . E8 AB0B0000   CALL   <JMP.&api-ms-win-crt-string-l1-    strcat
008810F2  . 8D85 7CFFFFFF LEA    EAX, DWORD PTR SS:[EBP-84]
Stack address=001BFA04, (ASCII "C:\Users\sysadmin\Documen")
EAX=001BFA04, (ASCII "C:\Users\sysadmin\Documen")
```

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.

Lab 6
Gunnar Yonker

```
008810F8  . 68 44218800   PUSH    Lab6_sam.00882144          src = "ts\"
008810FD  . 50            PUSH    EAX                        dest
008810FE  . E8 9A0B0000   CALL    <JMP.&api-ms-win-crt-string-l1-  strcat
00881103  . 8D85 7CFFFFFF LEA     EAX, DWORD PTR SS:[EBP-84]
Stack address=001BFA04, (ASCII "C:\Users\sysadmin\Documents\")
EAX=001BFA04, (ASCII "C:\Users\sysadmin\Documents\")
```

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.

```
00881109  . 50            PUSH    EAX                        path
0088110A  . FF15 6C208800 CALL    DWORD PTR DS:[<&api-ms-win-crt-  _chdir
00881110  . 83C4 24       ADD     ESP, 24
ESP=001BF9D0

Address   Hex dump                ASCII
001BFA04  43 3A 5C 55 73 65 72 73 C:\Users
001BFA0C  5C 73 79 73 61 64 6D 69 \sysadmi
001BFA14  6E 5C 44 6F 63 75 6D 65 n\Docume
001BFA1C  6E 74 73 5C 00 00 00 00 nts\....
```

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”.

```
pFindFileData
FileName = "*.docx"
FindFirstFileA

001BF7C0  02 00 04 06 70 61 73 73  ....pass
001BF7C8  77 6F 72 64 73 2E 64 6F  words.do
001BF7D0  63 78 00 77 DE FF FF FF  cx.wÞÿÿÿ
```

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.

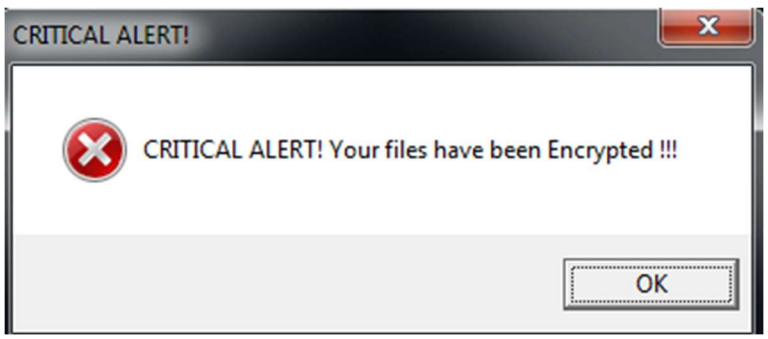
Lab 6
Gunnar Yonker

```
00881031  > 8D85 D8FDFFFF  LEA     EAX, DWORD PTR SS:[EBP-228]
00881037  . 50             PUSH    EAX
00881038  . FF15 00208800  CALL    DWORD PTR DS:[<&ADVAPI32.Encry  ADVAPI32.EncryptFileA
0088103E  . 8D85 ACFDFFFF  LEA     EAX, DWORD PTR SS:[EBP-254]
00881044  . 50             PUSH    EAX                             pFindFileData
00881045  . 56             PUSH    ESI                             hFile
00881046  . FF15 0C208800  CALL    DWORD PTR DS:[<&KERNEL32.FindN  FindNextFileA
0088104C  . 85C0           TEST    EAX, EAX
0088104E  .^75 E1          JNZ     SHORT Lab6_sam.00881031
00881050  . 56             PUSH    ESI                             hSearch
00881051  . FF15 14208800  CALL    DWORD PTR DS:[<&KERNEL32.FindCl FindClose
```

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.

```
Style = MB_OK|MB_ICONHAND|MB_APPLMODAL
Title = "CRITICAL ALERT!"
Text = "CRITICAL ALERT! Your files have been Encrypted !!!"
hOwner = NULL
MessageBoxW
```



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.

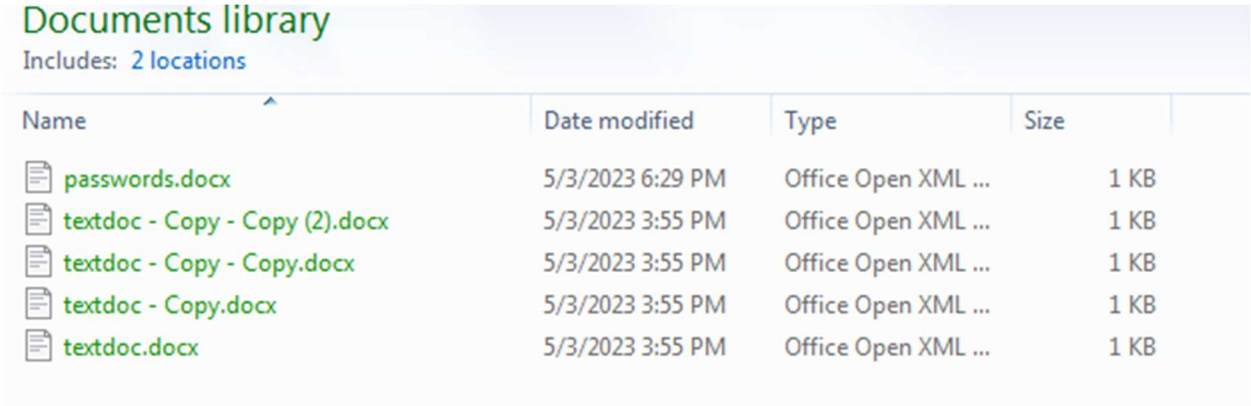
```
00881974  $ 6A 00          PUSH    0                               pModule = NULL
00881976  . FF15 4C208800  CALL    DWORD PTR DS:[<&KERNEL32.GetMod GetModuleHandleW
0088197C  . 85C0           TEST    EAX, EAX                        Lab6_sam.00880000
```

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.

```
00881387  > 56             PUSH    ESI                             status
00881388  . E8 46090000    CALL    <JMP.&api-ms-win-crt-runtime-11 exit
```

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

Lab 6
Gunnar Yonker



Documents library
Includes: 2 locations

Name	Date modified	Type	Size
passwords.docx	5/3/2023 6:29 PM	Office Open XML ...	1 KB
textdoc - Copy - Copy (2).docx	5/3/2023 3:55 PM	Office Open XML ...	1 KB
textdoc - Copy - Copy.docx	5/3/2023 3:55 PM	Office Open XML ...	1 KB
textdoc - Copy.docx	5/3/2023 3:55 PM	Office Open XML ...	1 KB
textdoc.docx	5/3/2023 3:55 PM	Office Open XML ...	1 KB

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.