**CSS 579: Malware Reverse Engineering**

**Team Malware Project**

**Malware Details**

*Binary*: winword64.exe (this is the only file that the other team needs for analysis)

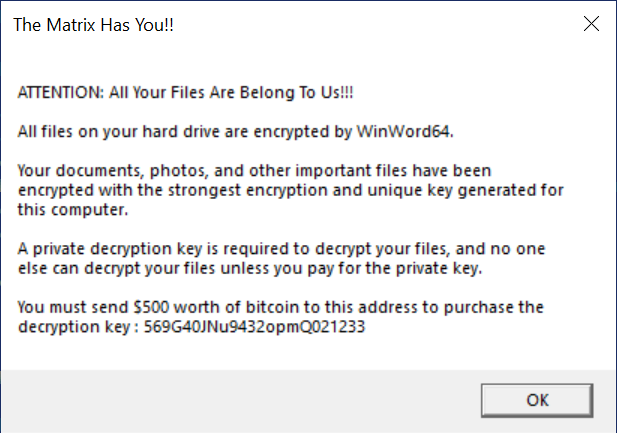
*Source Code*: winword64.cpp

*Type*: Ransomware

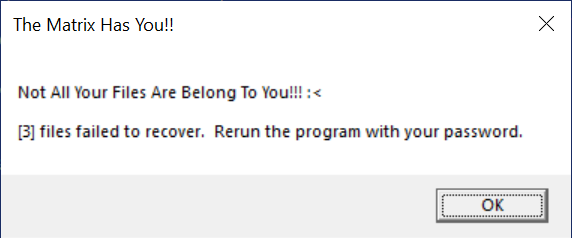
*Overview*: Winword64.exe is a Windows ransomware malware that encrypts files in the SYSTEMDRIVE:\Users directory with particular extensions targeting common image, video, audio, and document files.

*Techniques Implemented*:

1. **String Obfuscation**: All strings within the program are encoded in a reverse Hex encoding to obfuscate the strings during static analysis.
2. **Anti Analysis**: The program begins by performing an analysis check to determine if it is being debugged or running in a VM. If the analysis check is true, then the program exits. The program checks for No Pill and IsDebuggerPresent().
3. **Runtime String Deobfuscation**: The program then decodes all of its strings into real, readable, strings.
4. The program processes command line arguments to determine its execution mode - encrypt (the default with no args), or decrypt with a key.
5. If **Encrypting** files:
   * Uses the SHGetKnownFolderPath() API to find the User Profiles directory (c:\users on Windows 10, c:\documents and settings on Windows XP and older)
   * Searches the full Users directory for 21 file extensions targeting common image, video, audio, and document files.
   * Foreach file found:
     + Uses the CreateFileW() API to get a handle to the file, and a handle to a new file for storing the encrypted version.
     + Uses the APIs CryptAcquireContextW(), CryptCreateHash(), CryptHashData(), CryptDeriveKey(), ReadFile(), CryptEncrypt(), WriteFile(), CryptReleaseContext(), CryptDestroyKey(), and CryptDestroyHash() to create an encrypted version of each file.
     + All encrypted files are named using the original filename with a .encrypted file extension. For example, Image10.png would become Image10.png.encrypted.
     + After successful encryption, the original file is deleted.
   * **Persistence**: The program sets persistence by writing data to the Registry into these Keys using the GetModuleFileNameW(), RegCreateKeyExW(), RegSetValueExW(), and RegCloseKey():
     + Uses the GetModuleFileNameW() API to get the file path to the malicious executable.
     + Uses the APIs RegCreateKeyExW(), RegSetValueExW(), and RegCloseKey() to create or open the persistence key, Software\\Microsoft\\Windows\\CurrentVersion\\Run, in the Registry. Then it sets the name of the value in the Registry to WinWord64, and the value of it to the path of the malicious executable.
     + Uses the RegCloseKey() to close the key.
   * **Beaconing**: The program sends a beacon message to a Slack channel to report that an infection has completed. The beacon message includes the machine name, current username, date/time, and number of files encrypted. The networking APIs used for the beacon message are WinHttpOpen(), WinHttpConnect(), WinHttpOpenRequest(), WinHttpSendRequest(), and WinHttpCloseHandle(). Additional APIs used in the creation of the beacon message are GetComputerNameEx(), GetUserName(), and gmtime\_s(). Here is a sample of the Slack beacon:  
       
     New Infection! Computer: Spectrum; User: chris; Files Infected: 75; Timestamp: 2019-05-24 13:18:43 UTC
   * **Ransom Message**: Finally, the program displays a ransom message to the user.



1. If **Decrypting** files:
   * The program removes the persistence values from the Registry by deleting values using the APIs RegOpenKeyExW(), RegDeleteValue(), and RegCloseKey():
     + Uses the RegOpenKeyExW() API to open the key to the persistence key, WinWord64.
     + Uses the RegDeleteValue() API to delete the value from the key.
     + Uses the RegCloseKey() to close the key.
   * Searches the full Users directory for all files with .encrypted extensions.
   * Foreach file found:
     + Uses the CreateFileW() API to get a handle to the encrypted file, and a handle to a new file for storing the decrypted version.
     + Uses the APIs CryptAcquireContextW(), CryptCreateHash(), CryptHashData(), CryptDeriveKey(), ReadFile(), CryptDecrypt(), WriteFile(), CryptReleaseContext(), CryptDestroyKey(), and CryptDestroyHash() to create a decrypted version of each file to restore the original file.
     + After successful decryption, the .encrypted file is deleted.
   * **Beaconing**: The program sends a beacon message to a Slack channel to report that an infection has completed. The beacon message includes the machine name, current username, date/time, and number of files decrypted. The networking APIs used for the beacon message are WinHttpOpen(), WinHttpConnect(), WinHttpOpenRequest(), WinHttpSendRequest(), and WinHttpCloseHandle(). Additional APIs used in the creation of the beacon message are GetComputerNameEx(), GetUserName(), and gmtime\_s(). Here is a sample of the Slack beacon:  
       
     Cleaning Infection on Computer: REMWin10; User: REM; Files Decrypted: All; Timestamp: 2019-05-29 04:49:45 UTC
   * Finally, the program displays a message to the user letting them know that their files have been restored.
     + If not all of the files decrypted successfully, then the user is presented with this message instructing the user to rerun the program:



* + - If all files are decrypted successfully, then the user is presented with this message:

