MiTeC Exe Explorer:

Graphical user interface, text, application, email

Description automatically generated

Potentially Malicious Calls located in WININET.dll, typically programs do not need to connect to the internet and use InternetReadFile.

Also in KERNEL32.dll:

Graphical user interface, text, application

Description automatically generated

Sleep is potentially malicious and so is DeleteFileA, this does not guarantee it is malware but it could be.

Ghidra Analysis:

Graphical user interface

Description automatically generated with low confidence

The main function was located by searching for InternetGetConnectedState and then using the function call tree to find the correct main function. The param\_1 variable is pushed during the main function which is a value that we do not currently know. Once the main function takes place and establishes the stack, it moves onto the while loop.

Graphical user interface

Description automatically generated with medium confidence

The while\_loop label is returned back to often after call functions depending on the returned value, which I will outline further in the conclusion. An important piece of information to highlight here is that the loop relies on the value that is located in and incremented in the eax register. When the XOR EAX,EAX command takes place it is set to zero and then incremented by 1, so it moves past the jump and continues on. The program calls to internet\_connect\_status to find out if the system is connected to the internet or not.

A picture containing table

Description automatically generated

In this function a call to InternetGetConnectedState is made and this will return a value of 1 if there is a connection and 0 if there is no connection. This value will dictate what takes place next. If the internet connected state returns a value of 1 then the message pushed and displayed using thunk\_printf is “Downloading updates for Microsoft Edge…”. Then the eax register is cleared and incremented by 1 before returning to the while\_loop. If there is no internet connection then the jump to the label no\_active\_internet will take place.

Graphical user interface

Description automatically generated with medium confidence

If this jump is made, then the error message “Connection Error! Cannot download updates for Microsoft” will be displayed before the eax register is cleared and then returned to the while\_loop with a return value of 0.

Graphical user interface

Description automatically generated with medium confidence

The done register is jumped to once the internet connection state is determined and this is used to clean the stack and then return to the while\_loop.

Graphical user interface, text, application

Description automatically generated

If the value returned by the internet\_connect\_status shows that there is no active internet connection then the jump to label no\_active\_internet\_sleep will take place.

Graphical user interface

Description automatically generated with low confidence

The hex value of 0x36ee80 is pushed to the stack and a call to Sleep is made, the program then sleeps for 3600000 milliseconds before jumping back to the beginning of the while\_loop.

If there is an active internet connection then the call to thunk\_get\_url\_bytes is made and that thunk jumps to get\_url\_bytes.

Table

Description automatically generated

The call to InternetOpenW is made with the arguments shown, the most notable being that the application being used is Microsoft Edge. If the call is successful then the session\_handle is stored as a non-zero variable and the jump to internet\_handle\_success is made. If it is unsuccessful then the eax register is cleared and the jump to get\_url\_bytes\_done is made.

Graphical user interface, application

Description automatically generated

This returns the program to the while\_loop.

Table

Description automatically generated

In this label the call to InternetOpenUrlW is made, the argument to highlight is that the variable with the url is real\_url\_address(formerly labeled as param\_1), which is not the variable that was originally shown in the while\_loop. If this call is successful then the url handle is stored in the variable url\_handle and the jump to the internet\_read\_file is made. If this call is unsuccessful, then the eax register is cleared and the jump to the label get\_url\_bytes\_done is made as shown before.

Table

Description automatically generated with medium confidence

The internet\_read\_file label makes a call with InternetReadFile where the buffer is made to contain the command that is being read from the url. If this is successful then the command is stored in the eax register and a jump to read\_file\_successful is made. Otherwise, a jump to get\_url\_bytes\_done is made after the eax register is cleared.

Table

Description automatically generated with low confidence

The read command is stored for later use in the AL register from the buffer. Then the internet connection and the url connection are both closed before clearing the eax register and incrementing it by 1, and returning to the while\_loop.

Table

Description automatically generated with medium confidence

In this screenshot the continuation of the while\_loop is shown and if the get\_url\_bytes call was successful the call to thunk\_get\_url\_bytes\_success takes place.

Graphical user interface, text

Description automatically generated with medium confidence

Otherwise, the jump to get\_url\_byte\_failed takes place and a call to sleep for 604800000 milliseconds.

This next screenshot shows the intent that the program was trying to carry out if the while\_loop is successful in opening an internet connection, url connection, and receiving the command from the file read.

Table

Description automatically generated with medium confidence

The value contained in param\_1 that was read from the file read call earlier, is converted from a char to an int value using MOVSX and that value is contained in the eax register. This is compared to the hex value of 0x64 which is an int of 100 and corresponds to the letter “d”. If the value contained in param\_1 matches the letter “d” then "C:WindowsSystem32\ntdll.dll" is pushed to the stack and DeleteFileA is called. Thus this would delete that file and then jump to command\_done.

A picture containing timeline

Description automatically generated

If the param\_1 value when converted does not match 100, then the program jumps to param\_1\_not\_d. Then the value is converted again and compared to hex 0x73, 113 in decimal, ASCII the letter “s”. If this matches then the program pushes values to the stack for a shutdown and calls ExitWindowsEx. It is important to look at the arguments used for this call, uFlags being 0x4 and corresponding to EWX\_FORCE (used in emergency, can cause applications to lose data, force shutdown) and dwReason being zero and corresponding to the reason code is not set and logged at "No title for this reason could be found". Then the program jumps to label command\_done. If param\_1 does not match the letter “s”, then the program jumps to the label param\_1\_not\_s.

Timeline

Description automatically generated with medium confidence

In this case the value of 0x5265c00 is pushed and then the call to Sleep is made and the program sleeps for 86400000 milliseconds.

The label command\_done is also shown here where ebp is popped off the stack and then the program returns to the while loop.

Graphical user interface

Description automatically generated

Upon return the line param\_1 is popped off of the stack and then the program jumps to the label no\_active\_internet\_sleep. The call to Sleep is made and the program sleeps for 3600000 milliseconds before jumping back to the while\_loop label.

The label main\_done is jumped to if the eax value at the beginning of the while\_loop is zero.

Graphical user interface, application

Description automatically generated

**Conclusion:**

The malware appears to perform the following malicious activities:

The main function initializes the stack and then the program moves into a while loop. The while loop checks for active internet connectivity using the InternetGetConnectedState function. It is also at this point in the ASM code that the code tries to make the user believe that the url that is going to be accessed, if the code is looked at, belongs to microsoftupdates.com. If there is no active internet connection the error message “Connection Error! Cannot download updates for Microsoft Edge” is displayed and the malware sleeps for about an hour (3600000 milliseconds) before continuing to the next iteration of the while loop. If there is an active internet connection, the malware will display a success message, “Downloading updates for Microsoft Edge…” and then the malware will attempt to open a handle to malicious url. The call functions that the malware uses are InternetOpenW and InternetOpenUrlW to establish this session handles. If successful, the malware then attempts to call to InternetReadFile to read a command from the url connection. The data is written to a buffer and then the handles are closed before returning back to the while loop function. The command that was received from the InternetReadFile call is then checked before performing the next action. If the command received corresponds to the ASCII letter “d” then a file is deleted, specifically the call DeleteFileA is used to delete "C:WindowsSystem32/ntdll.dll". If the command received corresponds to the ASCII letter “s” then the malware will attempt a forceful shutdown of the system by calling ExitWindowsEx with the arguments that specific a EWX\_FORCE shutdown and that the dwReason code is logged as “No title for this reason could be found”. If neither of these values match, then the malware will call to another Sleep function and sleep for 86400000 milliseconds. Unless a system shutdown occurs, the program will then return back to the while loop and continue another iteration.

I believe that this malware attempts to make the user believe that it is downloading an update for Microsoft Edge, and then tries to enforce this belief by pushing the url of the microsoftupdates.com address in the code, but that url is never used in the call. There is a different url that is pushed and used in the call function when the handle is established that then allows the program to receive the intended data from that url. The program then has the goal of deleting the "C:WindowsSystem32/ntdll.dll" file or forcefully shutting down the system based on what command is received. The malware has sleep functions built in so that the while loop can continue its iterations at different times, constantly checking for there to be an internet connection. Once an internet connection is found, or the user believes that the program is attempting a Microsoft Edge update so they connect to the internet, the program will carry out with deleting the file or forcefully shutting down the system.