Web Security and Malware Analysis

Answers for Assignment 8

SILVIA LUCIA SANNA – 70/90/00053

Task 1 – Full Malware Analysis

In this task I have to make a full as possible analysis of this malware.

Looking at the PE structure I can recognise it is a MS-DOS file executable for a 32-bit machine with Intel 386 and specific OS versions (majorOS and minorOS have values respectively 4 and 0 but I did not find to which versions correspond). Analysing the sizes, I can easily see that there is a lot of code (size of code is 0xA000) and enough data (size of data is 0x6000) but no uninitialized data. From base of code and base of data, combining them with image base, I have the address where code and data starts. In imported directory I have all imported libraries: kernel32.dll, advapi32.dll, shell32.dll, WS2_32.dll.

After this I can analyse the strings and make a little backward approach of static analysis. Analysing the strings found in .data section I see that some functions are repeated, so I can start analysing them.

.data:0040C030	0000000E	C	Configuration
.data:0040C040	00000018	C	SOFTWARE\\Microsoft \\XPS
.data:0040C058	0000000E	C	\\kernel32.dll
.data:0040C070	0000000E	C	HTTP/1.0\r\n\r\n
.data:0040C088	00000006	C	`%s`
.data:0040C090	00000006	C	`%s`
.data:0040C098	00000008	C	NOTHING
.data:0040C0AC	00000009	C	DOWNLOAD
.data:0040C0B8	00000007	C	UPLOAD
.data:0040C0C4	00000006	C	SLEEP
.data:0040C0CC	00000008	C	cmd.exe
.data:0040C0D4	00000008	C	>> NUL
.data:0040C0DC	00000008	C	/c del
.data:0040C0E8	00000028	C	http://www.practicalmalwareanalysis.com
.data:0040C118	00000011	C	Manager Service
.data:0040C134	00000017	C	%SYSTEMROOT%\\system32\\
.data:0040C14C	00000017	C	k:%s h:%s p:%s per:%s\n

Figure 1: .data section in Strings window

- 401000: a request to a specific value of a register is made, which is the value "Configuration" of "Software\Microsoft\XPS" register.
- 401070: a concatenation with input values is made and this is used to create a new value of Configuration in Software\Microsoft\XPS register.
- 4015B0: opens kernel32.dll in system directory and copies its time value (creation, last modification and last access) to the file passed as input.
- 401AF0: calls 401640 (performs connection to specific host) send some get requests, waiting the responses and qmemcopy.
- 401E60: a complex function composed by different functions which retrieves some values from the register "Software\Microsoft\XPS" register value "Configuration" and then calls 401AF0.
- 402020: performs different actions depending on the input string, like nothing, "upload", "download", "cmd" or str2 (I will explain later what happens in detail).
- 402410 which deletes the current process, in fact a shell with \c /del is called.
- 402600: a function calling different functions like 4015B0. At the end creates a key with a url value (I will explain later what happens in detail).
- 402900: a complex function with different functions calls that briefly delete some file (I will explain later what happens in detail).

Some of the info reported here can also be found analysing other strings in String Window like the following ones.

.rdata:0040B7EA	0000000C	C	CloseHandle
.rdata:0040B7F8	0000000C	C	SetFileTime
.rdata:0040B806	0000000C	C	GetFileTime
.rdata:0040B814	0000000C	C	CreateFileA
.rdata:0040B822	00000014	C	GetSystemDirectoryA
.rdata:0040B838	0000000D	C	GetLastError
.rdata:0040B848	00000009	C	ReadFile
.rdata:0040B854	0000000A	C	WriteFile
.rdata:0040B860	00000006	C	Sleep
.rdata:0040B868	00000012	C	GetShortPathNameA
.rdata:0040B87C	00000013	C	GetModuleFileNameA
.rdata:0040B892	0000000A	C	CopyFileA
.rdata:0040B89E	0000001A	C	ExpandEnvironmentStringsA

Figure 2: part of .rdata section in Strings window

Now I have to join this information and to do so I perform a forward static analysis, exploring what each function does and how they are linked.

In the static analysis I watched at the main function and explored every function and conditions. Using this method, I realised that if I pass nothing to it, or better if I execute it without parameters, it will delete and will modify some registers value. I also discovered that I have to insert a password (sub_402510) to execute this malware and it must be the last term. Analysing 402510 I know that the password is 'abcd', it is not a clear text password, but it is like a riddle. With the password I also have to insert different commands and parameters, where each combination makes a specific configuration. So, to execute this malware avoiding deleting itself by calling 402410, I can pass as input the following parameters where the first one will be always the name and the last one will be the password:

- Lab09-01.exe -cc abcd: it saves in 4 different variable (each one of 1024 bytes) the value of "Configuration" of "Software\Microsoft\XPS" register and it will print the first 1024 bytes. So maybe the mean of this command is to check the configuration.
- Lab09-01.exe -c p2 p3 p4 p5 abcd: it will set the value of "Configuration" in "Software\Microsoft\XPS" register with the concatenation of the 4 parameters passed as input (p2, p3, p4, p5). So, this command makes a configuration.
- Lab09-01.exe -re abcd or Lab09-01.exe -re p2 abcd: if p2 is not passed it calls function 4025B0 and puts the path of the running file in v8. Here function 402900 is called: first it makes a connection to the System Control Manager (SCM) and opens the service v8 or the one passed as input (p2). Then will delete this service and also delete the file found when the environment variable was expanded "%SYSTEMROOT%\system32\Lab09-01.exe" (this last part of the path, Lab09-01.exe is the name of the executable file obtained by calling function 4025B0), after that puts in

Configuration of Software\Microsoft\XPS the value “4444”. Here there is a problem: the service is opened but if p2 is passed, maybe the service with p2 is not created (or better I did not encountered any function before that creates the service and there is no function after that creates it if not present). Even with v8 maybe the service is not created: if I execute only “Lab09-01.exe -re abcd”, in function 402900 there is no service creation. I will discover later with the debug how this function works, but probably it must be called after some service creation with the aim to delete it. So the purpose of this command can be to remove a service (default or passed as input).

- Lab09-01.exe -in abcd or Lab09-01.exe p2 abcd: here again as in 402900 if p2 is not passed, it will call 4025B0 to put the path of the running file in v8 variable. Then it will connect to SCM and open the service with v8 or p2. Here, unlike 402900, there is a check if the opening works or not. So, if the opening works, tries to change some configurations but if it fails close everything and ends with exit code 1. Otherwise, if does not open the service, this means that does not exist, so creates it (if has some problems while creating, close the connection and returns 1). After this, will expand the environment variable, get the file path, copy this file in the environment expanded and then call 4015B0 (I guess that sets creation time / modification time and last access time to the file in the expanded environment – which is the copy of the running file – with the times of kernel32.dll in the system directory). This is a way to hide itself. After that will set the value of Configuration in Software\Microsoft\XPS to a specific URL composed by 4 different fields “ups”, “https://practicalmalwareanalysis.com”, “80”, “60”, the first option in the program to make a network connection. So with this command you maybe can install a service, hide the executable in other path and set a key to connect to the network.

If other options except these ones are passed, the program will delete itself.

So, analysing these points, I can confirm that the parameter -re can be run only after a service is created to delete it, otherwise it will not delete anything and close connections for unsuccess opening. To be sure of this result, I must debug 402900.

The static analysis is not finished because there is the part regarding function 402360, the one executed if no parameters are passed.

If the executable file is executed without any parameter, it retrieves the value of Configuration in Software\Microsoft\XPS (function 401000) and delete the executable (function 402410). After that, but also if is not able to retrieve the Configuration value, it calls function 402360 that I am going to analyse.

```
1  int sub_402360()
2  {
3    int v1; // eax
4    char String[1024]; // [esp+0h] [ebp-1000h] BYREF
5    char v3[1024]; // [esp+400h] [ebp-C00h] BYREF
6    char name[1024]; // [esp+800h] [ebp-800h] BYREF
7    char v5[1024]; // [esp+C00h] [ebp-400h] BYREF
8
9    while ( 1 )
10   {
11     if ( sub_401280(v3, 1024, name, 1024, String, 1024, v5) )
12       return 1;
13     atoi(String);
14     if ( sub_402020(name) )
15       break;
16     v1 = atoi(v5);
17     Sleep(1000 * v1);
18   }
19   return 1;
20 }
```

Figure 3: function 402360

As we can see in line 9, this is a forever loop where only 4 functions are called:

- 401280: retrieves the value Configuration of register Software\Microsoft\XPS storing in variables v3, name, String, v5, each one of 1024 bytes.

- Atoi: converts the variable String (values of Configuration in that register from 2048 to 3071 bytes) from string type to integer type.
- 402020: this function is a bit complex and does different things. First of all calls 401E60 which opens winsock.dll and creates a socket connection with specific parameters (function 401640 inside function 401AF0), with hostname composed using 1024-2047 bytes of Configuration value in register Software\Microsoft\XPS and hostshort value 2048 of same register field Configuration; after this sends via socket a Get request using HTTP1 to a casual string generated in function 401D80; then receives the response storing at least 4096 bytes but breaking if there is a specific substring (in hex 0D0A). The function 401E60 will return Str1 which is the fifth string of the array of occurrences of substring asc_40C090. After this, in 402020 a check of different types of substrings in Str1 is made:
  - If is equal to Str2 (which I am not able to know what is, I supposed is the name variable passed in input to 402020, to know it I think that I must debug): the process will sleep for a certain number of milliseconds.
  - If is “Upload”: a socket is created and also a file in which all bytes received (512 at a time) by the sockets are written and, when it ends, the socket and the file are closed.
  - If is “Download”: a socket is created, a file is opened and all its bytes are sent in the socket (512 at a time). At the end, the socket and the file are closed.
  - If is “CMD”: a pipe in read binary mode is created and the flow is read, then a socket is created and all read bytes in the pipe are sent in the socket. When there are no more bytes to be sent, the socket is closed and also the pipe.
  - If is “Nothing”: the program finishes and continues its flow in function 402360.
- Sleep: in this function the program will sleep for a certain number of milliseconds, which is the integer value of the

last 1024 bytes of Configuration value in register Software\Microsoft\XPS.

In this part too, there are some variables that I do not know their value so I must debug the program to retrieve it, for example the values of files to be opened and in which to write. Using the socket, this is the second part where a network connection is established.

So, if no parameters are passed to the program, it will randomly Upload, Download and execute a pipe in a forever loop, I think until there is space in the disk or until string "Nothing" has been randomly generated. So, if this is right, I can classify it as a classical virus that destroys the target machine (which I remember must be a 32 bit) by filling in all available space in disk.

In all this program, for any kind of error, the returned value is 1.

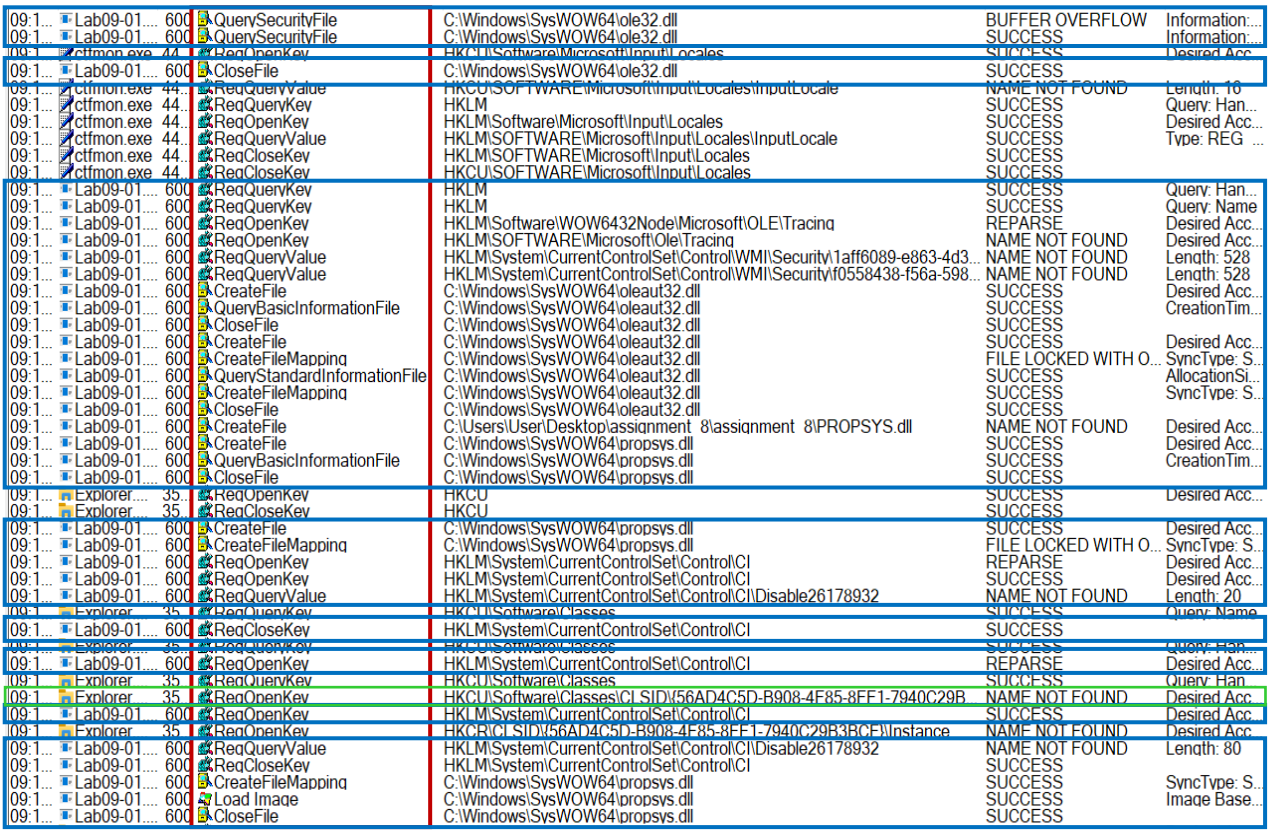
So, a brief recap of how this program works. To execute it I can send 4 different types of commands but the third one (-re) only after a first execution with -in command, because there is the opening of a specific service set in other functions (I suppose this is function 402600 when calls CreateServiceA at line 33). So, in my opinion the first time we have to execute this program there are 3 different options: Lab09-01.exe -cc abcd; Lab09-01.exe -c p2 p3 p4 p5 abcd; Lab09-01.exe -in abcd.

Then you can execute this program other times using: Lab09-01.exe -re abcd or without parameters so only Lab09-01.exe.

To confirm all the hypothesis I made in the static analysis, I have to perform some dynamic analysis.

I decided to run the program without any parameter (the program deleted itself) and I opened "procmon" to monitor what happens for every process: I filtered for "Lab09-01.exe" and I saw a lot of operations with registers and files. This means that function 401000 is called successfully and then the random functions for Upload, Download and CMD are called (in fact there are functions to open and close file). What I noticed is that the path changes

a lot and these values in my opinion can be what I previously named as, for example, first 1024 bytes of value Configuration in register Software\Microsoft\XPS.



09:1...	Lab09-01....	600	QuerySecurityFile	C:\Windows\SysWOW64\ole32.dll	BUFFER OVERFLOW	Information:...
09:1...	Lab09-01....	600	QuerySecurityFile	C:\Windows\SysWOW64\ole32.dll	SUCCESS	Information:...
09:1...	ctfmon.exe	44...	RegOpenKey	HKCU\Software\Microsoft\Input\Locales	SUCCESS	Desired Acc...
09:1...	Lab09-01....	600	CloseFile	C:\Windows\SysWOW64\ole32.dll	SUCCESS	
09:1...	ctfmon.exe	44...	RegQueryValue	HKCU\SOFTWARE\Microsoft\Input\Locales\InputLocale	NAME NOT FOUND	Length: 16
09:1...	ctfmon.exe	44...	RegQueryKey	HKLM	SUCCESS	Query: Han...
09:1...	ctfmon.exe	44...	RegOpenKey	HKLM\Software\Microsoft\Input\Locales	SUCCESS	Desired Acc...
09:1...	ctfmon.exe	44...	RegQueryValue	HKLM\SOFTWARE\Microsoft\Input\Locales\InputLocale	SUCCESS	Type: REG_...
09:1...	ctfmon.exe	44...	RegCloseKey	HKLM\SOFTWARE\Microsoft\Input\Locales	SUCCESS	
09:1...	ctfmon.exe	44...	RegCloseKey	HKCU\SOFTWARE\Microsoft\Input\Locales	SUCCESS	
09:1...	Lab09-01....	600	RegQueryKey	HKLM	SUCCESS	Query: Han...
09:1...	Lab09-01....	600	RegQueryKey	HKLM	SUCCESS	Query: Name
09:1...	Lab09-01....	600	RegOpenKey	HKLM\Software\WOW6432Node\Microsoft\OLE\Tracing	REPARSE	Desired Acc...
09:1...	Lab09-01....	600	RegOpenKey	HKLM\SOFTWARE\Microsoft\Ole\Tracing	NAME NOT FOUND	Desired Acc...
09:1...	Lab09-01....	600	RegQueryValue	HKLM\System\CurrentControlSet\Control\WMI\Security\1aff6089-e863-4d3...	NAME NOT FOUND	Length: 528
09:1...	Lab09-01....	600	RegQueryValue	HKLM\System\CurrentControlSet\Control\WMI\Security\f0558438-f56a-598...	NAME NOT FOUND	Length: 528
09:1...	Lab09-01....	600	CreateFile	C:\Windows\SysWOW64\oleaut32.dll	SUCCESS	Desired Acc...
09:1...	Lab09-01....	600	QueryBasicInformationFile	C:\Windows\SysWOW64\oleaut32.dll	SUCCESS	CreationTim...
09:1...	Lab09-01....	600	CloseFile	C:\Windows\SysWOW64\oleaut32.dll	SUCCESS	
09:1...	Lab09-01....	600	CreateFile	C:\Windows\SysWOW64\oleaut32.dll	SUCCESS	Desired Acc...
09:1...	Lab09-01....	600	CreateFileMapping	C:\Windows\SysWOW64\oleaut32.dll	FILE LOCKED WITH O...	SyncType: S...
09:1...	Lab09-01....	600	QueryStandardInformationFile	C:\Windows\SysWOW64\oleaut32.dll	SUCCESS	AllocationSi...
09:1...	Lab09-01....	600	CreateFileMapping	C:\Windows\SysWOW64\oleaut32.dll	SUCCESS	SyncType: S...
09:1...	Lab09-01....	600	CloseFile	C:\Windows\SysWOW64\oleaut32.dll	SUCCESS	
09:1...	Lab09-01....	600	CreateFile	C:\Users\User\Desktop\assignment_8\assignment_8\PROPSYS.dll	NAME NOT FOUND	Desired Acc...
09:1...	Lab09-01....	600	CreateFile	C:\Windows\SysWOW64\propsys.dll	SUCCESS	Desired Acc...
09:1...	Lab09-01....	600	QueryBasicInformationFile	C:\Windows\SysWOW64\propsys.dll	SUCCESS	CreationTim...
09:1...	Lab09-01....	600	CloseFile	C:\Windows\SysWOW64\propsys.dll	SUCCESS	
09:1...	Explorer....	35...	RegOpenKey	HKCU	SUCCESS	Desired Acc...
09:1...	Explorer....	35...	RegCloseKey	HKCU	SUCCESS	
09:1...	Lab09-01....	600	CreateFile	C:\Windows\SysWOW64\propsys.dll	SUCCESS	Desired Acc...
09:1...	Lab09-01....	600	CreateFileMapping	C:\Windows\SysWOW64\propsys.dll	FILE LOCKED WITH O...	SyncType: S...
09:1...	Lab09-01....	600	RegOpenKey	HKLM\System\CurrentControlSet\Control\CI	REPARSE	Desired Acc...
09:1...	Lab09-01....	600	RegOpenKey	HKLM\System\CurrentControlSet\Control\CI	SUCCESS	Desired Acc...
09:1...	Lab09-01....	600	RegQueryValue	HKLM\System\CurrentControlSet\Control\CI\Disable26178932	NAME NOT FOUND	Length: 20
09:1...	Explorer....	35...	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: Name
09:1...	Lab09-01....	600	RegCloseKey	HKLM\System\CurrentControlSet\Control\CI	SUCCESS	
09:1...	Explorer....	35...	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: Han...
09:1...	Lab09-01....	600	RegOpenKey	HKLM\System\CurrentControlSet\Control\CI	REPARSE	Desired Acc...
09:1...	Explorer....	35...	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: Han...
09:1...	Explorer....	35...	RegOpenKey	HKCU\Software\Classes\CLSID\{56AD4C5D-B908-4E85-8FF1-7940C29B...	NAME NOT FOUND	Desired Acc...
09:1...	Lab09-01....	600	RegOpenKey	HKLM\System\CurrentControlSet\Control\CI	SUCCESS	Desired Acc...
09:1...	Explorer....	35...	RegOpenKey	HKCR\CLSID\{56AD4C5D-B908-4E85-8FF1-7940C29B3BCE}\Instance	NAME NOT FOUND	Desired Acc...
09:1...	Lab09-01....	600	RegQueryValue	HKLM\System\CurrentControlSet\Control\CI\Disable26178932	NAME NOT FOUND	Length: 80
09:1...	Lab09-01....	600	RegCloseKey	HKLM\System\CurrentControlSet\Control\CI	SUCCESS	
09:1...	Lab09-01....	600	CreateFileMapping	C:\Windows\SysWOW64\propsys.dll	SUCCESS	SyncType: S...
09:1...	Lab09-01....	600	Load Image	C:\Windows\SysWOW64\propsys.dll	SUCCESS	Image Base...
09:1...	Lab09-01....	600	CloseFile	C:\Windows\SysWOW64\propsys.dll	SUCCESS	

Figure 4: snapshot of procmon where in the column with the red box there are the operation made. In the horizontal blue box are highlighted info about Lab09-01 process. Also Explorer process in my opinion is important because maybe asks for files (as in the green box where there is "Name not found" error) or interacts with it to put a file in a specific folder

Using regshot, I can have an automatic analysis of keys and register before and after execution: I took a shot on regshot when I turn on the snap-shot of the VM, I saved it, I run the program with the different parameters (for every parameter I used a clean snapshot of the VM to not have the registers influenced or changed by previous parameters), I saved it and compared the two savings so before and after parameter execution. For example, if I execute the program without parameters: 31 keys are created, 93 value added and 48 values modified, 13 files added, 3 files deleted, 31 files modified, 1 folder added and 1 folder attribute changed (which is "C:"). Here, as I thought that without parameters the loop was infinite, I don't know why regshot was able to make this analysis and in few minutes: if the loop is infinite there should be no fixed result

as instead I have. To prove this part I decided after hours of the execution without parameters to make another shot and compare the one generated as soon as the program is launched without parameters and the shot after hours without doing anything with the computer. Results are reported in the table above.

Again without any passed parameter I tried to use ApateDNS to discover which are the links to which the socket connects but I had only the following output. Here again I expected much more links as the requests are part of an infinite loop: Download, Upload and CMD (the functions that make socket connections) are called in 402020 which in turn called inside an infinite loop (while (1) in function 402360).

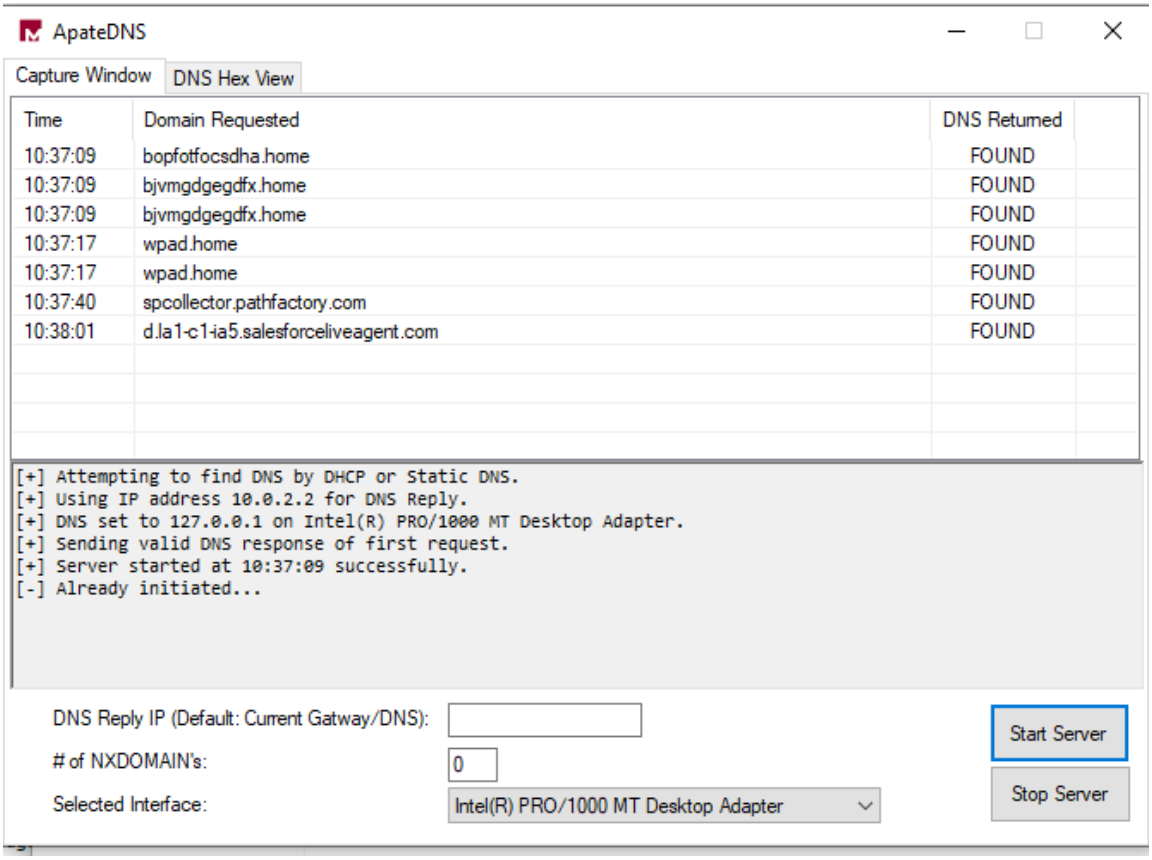


Figure 5: ApateDNS output

With regshot I analysed also how the registers change after I execute the program with different parameters (of course every command is executed with a clean status, to not have effects from previous commands):

Cmd	KD	KA	VD	VA	VM	FIA	FID	FIM	FOA	FOD	FOC
1	0	31	0	93	48	13	3	31	1	0	1
2	87515	139	251042	332	134	23	3	83	2	2	1
3	87472	33734	250953	100639	81	23	19	82	2	6	1
4	1	13	3	55	53	6	0	31	1	0	0
5	3	5	13	23	38	7	1	32	0	0	0
6	1718	5185	9532	29577	604	86623	92520	394	33386	30916	0

Table 1: in the first column there are the executed commands (1: no parameters, only double click on the executable; 2: -cc; 3: -in; 4: -in and -re, so the effects after -re are reported but this command must be run after -in; 5: -in and no parameters, so -in and after a simple double click on the executable; 6: no parameters so only double click on the executable and reporting results with respect to previous no parameters execution, so how registers change after few hours that the program is executed without parameters as it is in an infinite loop). Parameter "-c p2 p3 p4 p5 abcd" has not been executed as I did not know neither was able to recover the 4 parameters to launch. The following columns are composed in this way: Keys Deleted (KD), Keys Added (KA), Values Deleted (VD), Values Added (VA), Values Modified (VM), Files Added (FIA), Files Deleted (FID), Files Modified (FIM), Folders Added (FOA), Folders Deleted (FOD), Folders Changed (FOC). All shots have been scanned from directory C:\, so all directories have been analysed for every command.

Another interesting thing is that when I shut down the snap-shot in the Virtual Machine, after the execution without parameters, the machine took enough minutes to stop itself. Maybe it is only a coincidence or maybe because of the changes in the registers.

As from static and dynamic analysis I have some doubts about how some functions work, I thought to debug them:

- I would like to know the output of function "ExpandEnvironmentStringsA" in 402600 and in 402900. First I tried to debug it in 402900 but as I discovered in static analysis it blocks at function "OpenServiceA" which returns a NULL value so will not execute the if at line 14 but the else at line 56 which will close the connection returning 1 and so ending also the main function. This debug is executed with options "-re abcd".

```
1  BOOL __cdecl sub_402900(LPCSTR lpServiceName)
2  {
3    BOOL result; // eax
4    SC_HANDLE hService; // [esp+Ch] [ebp-C08h]
5    char v3[1024]; // [esp+10h] [ebp-C04h] BYREF
6    CHAR Dst[1024]; // [esp+410h] [ebp-804h] BYREF
7    SC_HANDLE hSCManager; // [esp+810h] [ebp-404h]
8    CHAR Src[1024]; // [esp+814h] [ebp-400h] BYREF
9
10   hSCManager = OpenSCManagerA(0, 0, 0xF003Fu);
11   if ( !hSCManager )
12     return 1;
13   hService = OpenServiceA(hSCManager, lpServiceName, 0xF01FFu);
14   if ( hService )
15   {
56   else
57   {
58     CloseServiceHandle(hSCManager);
59     result = 1;
60   }
61   return result;
62 }
```

Figure 6: execution flow of function 402900

So, the other method to discover what function "ExpandEnvironmentStringsA" does, is to debug function 402600. I started to debug it but at OpenSCManagerA it ends because it fails establishing the connection to the SCM on the local computer (first input of the function is 0 meaning local computer) and opening the SCM database (second input of the function is 0 and means to open by default the SERVICE_ACTIVE_DATABASE). Maybe it returns NULL because of the third parameter passed as input, which is different from the third parameter of same function in 402900. As the connection is not successful the program will return 1 and ends also the main. So, here again I cannot discover the output value of "ExpandEnvironmentStringsA". This debug is made with options "-in abcd". I also would like to discover how last part of 402600 works (lines 42-51), but I cannot reach it in debugging as returns 1 at line 18.

```
1  BOOL __cdecl sub_402600(LPCSTR lpServiceName)
2  {
3    SC_HANDLE hService; // [esp+Ch] [ebp-1408h]
4    char v3[1024]; // [esp+10h] [ebp-1404h] BYREF
5    CHAR Filename[1024]; // [esp+410h] [ebp-1004h] BYREF
6    CHAR DisplayName[1024]; // [esp+810h] [ebp-C04h] BYREF
7    CHAR BinaryPathName[1024]; // [esp+C10h] [ebp-804h] BYREF
8    SC_HANDLE hSCManager; // [esp+1010h] [ebp-404h]
9    CHAR Src[1024]; // [esp+1014h] [ebp-400h] BYREF
10
11   if ( sub_4025B0(v3) )
12     return 1;
13   strcpy(Src, aSystemrootSyst);
14   strcat(Src, v3);
15   strcat(Src, aExe);
16   hSCManager = OpenSCManagerA(0, 0, 0xF003Fu);
17   if ( !hSCManager )
18     return 1;
42   if ( !ExpandEnvironmentStringsA(Src, BinaryPathName, 0x400u) )
43     return 1;
44   if ( !GetModuleFileNameA(0, Filename, 0x400u) )
45     return 1;
46   if ( !CopyFileA(Filename, BinaryPathName, 0) )
47     return 1;
48   if ( sub_4015B0(BinaryPathName) )
49     return 1;
50   return sub_401070(aUps, aHttpWwwPractic, a80, a60) != 0;
51 }
```

Figure 7: execution flow of function 402600

Here a question is raised: if function 402600 is not able to make the connection and consequently not open or create the service, how can function 402900 work, or better how can -re command work? From this problem found in debugging, here I am not sure how the total program works as some functions are not executed entirely. I have two options: it only prints the first 1024 bytes of "Configuration" of register "Software\Microsoft\XPS" or set all its 4096 bytes. But this idea is incoherent with regshot output because for example after executing -in options and immediately after executing -re, some keys, files and folders change or are added or deleted. The second option is that I have to like patch the program in the debugging part to execute specific lines that at runtime are executed maybe because of specific checks or other things set only at runtime.

- The other thing I would like to discover is how the part regarding strtok works in function 402020, which is equal in all different "branches" (if the string contains Upload, Download, CMD or Str2) but changes only values.

```
26  else if ( !strncmp(Str1, aUpload, strlen(aUpload)) )
27  {
28    strtok(Str1, Delimiter);
29    v2 = strtok(0, Delimiter);
30    v10 = atoi(v2);
31    v11 = strtok(0, Delimiter);
32    if ( sub_4019E0(name, v10, v11) )
33      return 1;
34  }
```

Figure 8: part regarding strtok found in the second string comparison ("Upload") of 402020 function but equal to all other 3 string comparisons

This part cannot be debugged because it is part of function 402360 executed after 402410 which deletes the program and so the debugger stops. So, I cannot neither know the value of Str2 inside 402020 function. Here, maybe this can also happen at runtime and so probably 402020 is never called, but this hypothesis is in contrast with what I have found with procmon where I have different "Create File" and "Read File" when executing the program without parameters. So maybe this is only a debugging error.

At the end, I decided to use VirusTotal to know if my previous assumption about the nature of this malware is right. I supposed was a virus but VirusTotal says that is a backdoor/trojan. In fact this program uses socket and keeps a permanent connection, so it can be classified as a backdoor.

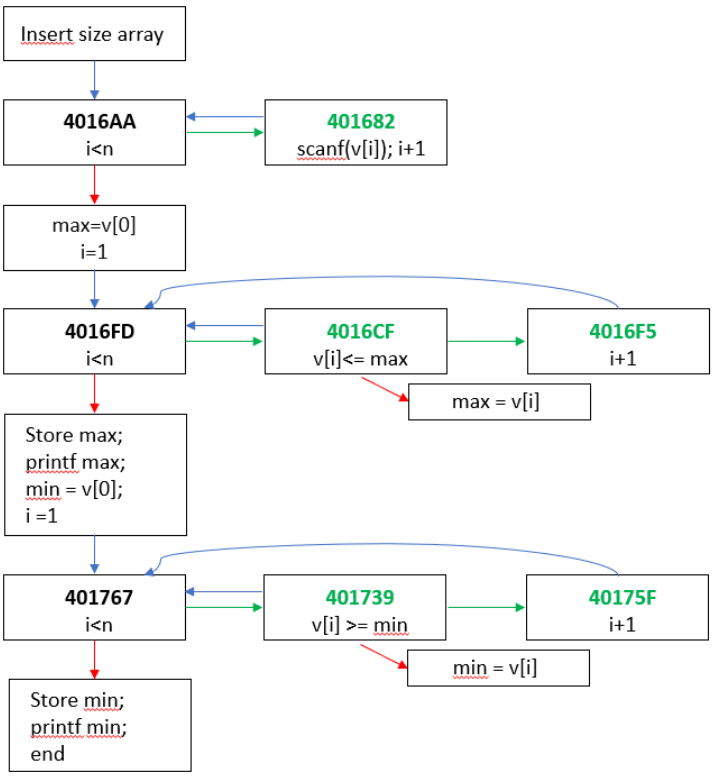
Task 2 – More Static Analysis

In this task a file with different lines of assembly is given to me and I have to analyse it.

In the first part there is the stack preparation, call to __main function and is asked to the user to insert the size of the array (stored in esp+18h and in esp+4h), then esp+0ECh is initialized to 0 which in C language means i=0. Immediately after this jump to 4016AA.

- 4016AA: the index is compared to the size array and if is less (i<n), jump to 401682, otherwise store the value in esp+1Ch (the first value of the array) in a new variable esp+0E8h which I called max (I explain in 4016CF why), the index is initialized to 1 (i=1) and jump immediately to 4016FD.
- 401682: here there is the scanf to fill in the array with the values inserted by the user and the index is incremented by 1 and comes back at the start of 4016AA to check if i<n.
- 4016FD: check if i<n and if yes jump to 4016CF, otherwise (i>n) store in a specific variable esp+4 the value found in esp+08Eh (which I called max, in the next point explain why); then print a value and store in a new variable esp+0E4h (which I called min, explain in 401739 why) the first value of the array (esp+1ECh); initialized the value of the index to 1 (esp+0ECh=1 which means in C i+=1) and jump immediately to 401767.
- 4016CF: compare the ith value of the array (esp+0ECh*4+1Ch) with max value and if is less or equal jump to 4016F5, otherwise esp+0E8h takes the value of v[i]. So esp+0E8h is variable max because takes the greatest value of the array, checking it at every iteration.
- 4016F5: index i is incremented by 1 and come back to first line of 4016FD to check if i<n.

- 401767: i<n is checked and if is correct, jump to 401739, otherwise (i>n) store in esp+4 the value of esp+0E4h; makes a printf and ends.
- 401739: a comparison between v[i] (esp+0ECh*4+1Ch, the ith value of the array) and esp+0E4h is made and if greater or equal jump to 40175F, otherwise esp+0E4h takes v[i] (so esp+0E4h is min variable because takes the smallest value of the array, checking it at every iteration with the ith value).
- 40175F: index i is incremented by 1 (i+=1) and comes back to first line of 401767 to check if i<n.




```c
#include <stdio.h>

int main() {
    // Write C code here
    int n=0;
    int v[100];
    printf("Enter size array \n");
    scanf("%d", &n);
    printf("Enter %d elements of the array \n", n);
    for (int i =0; i<n; i++){
        scanf("%d", &v[i]);
    }
    int max = v[0];
    for (int i=1; i<n; i++){
        if (v[i]>max){
            max=v[i];
    }
    printf("Result 1: \n", max);
    int min = v[0];
    for (int i=1; i<n; i++){
        if (v[i]<min){
            min=v[i];
    }
    printf("Result 1: \n", min);
    return 0;
}
```

Figure 9: on the left I made schema with the flow of the program in Ida style, containing the number of the location, what it does and arrows (red color is for branch not taken, green is for branch taken and blue for immediate jump). On the right there is the C code for the assembly code I analysed

So briefly this is the assembly of a program that checks the maximum and minimum numbers in an array.