Contents

1 Homework

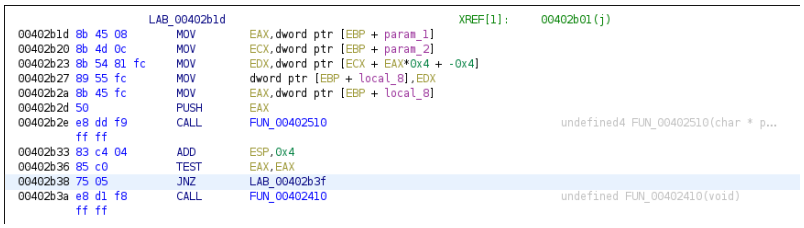
1.1 HW7

1.1.1 Lab 9-1

1. Question 1

   To get the malware to install, we need to reach `0x00402600`. Within this function, there are function calls to `OpenSCManagerA`, `ChangeServiceConfigA`, `CreateServiceA`, `CopyFileA`, and registry modifications.

   To get to the install function, we would need to run this malware with 3 arguments. We need a password as one of these arguments along with `"-in"` as the first argument. To decipher this password, we can take a look at `0x00402510`. The password must be 4 characters long. After analyzing the funciton, we see that the passcode is `"abcd"`.



   We can also patch `0x00402B38` by changing `jnz` to `jz` to bypass any password check.

   We can install the malware by executing it with the arguments `"Lab09-01.exe -in abcd"`.

2. Question 2

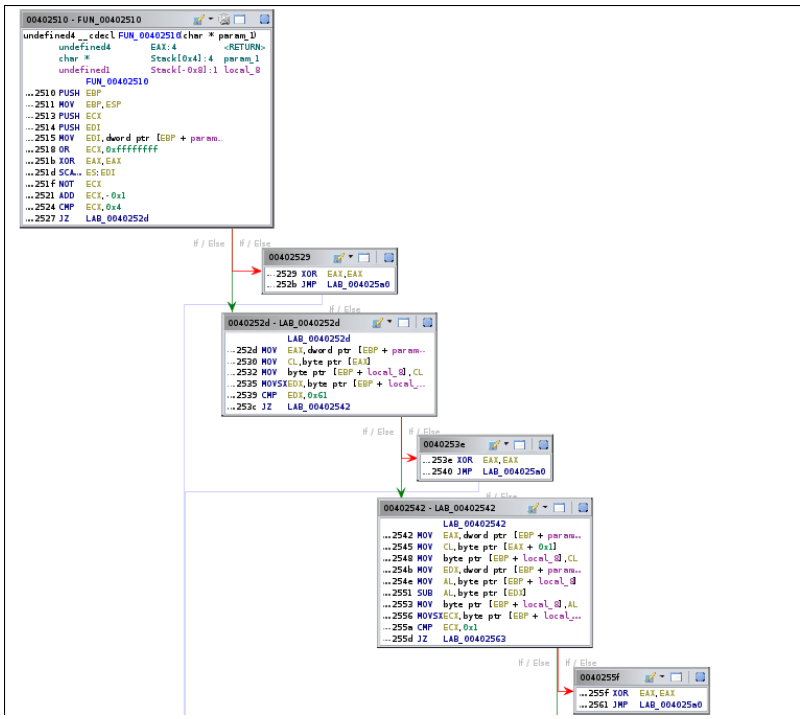
   There are 4 command-line options for the program.

   (a) `"-in"`: installs

(b) `"-re"`: uninstalls

(c) `"-cc"`: prints our registry

(d) `"-c"`: sets registry value



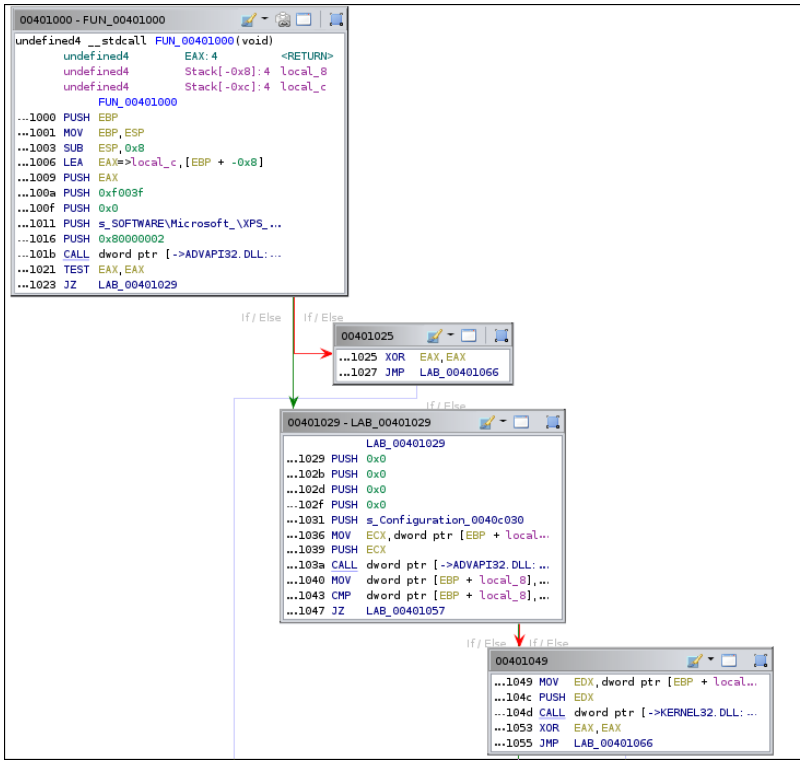
Analyzing the function, we can find what the password to the installer, `"abcd"`.

3. Question 3

   To Patch the program so that it doesnt require a password, we need to patch 0x00402B38 from `jnz` to `jz`.

Changing `75` to `74` will change the instruction from `jnz` to `jz`.

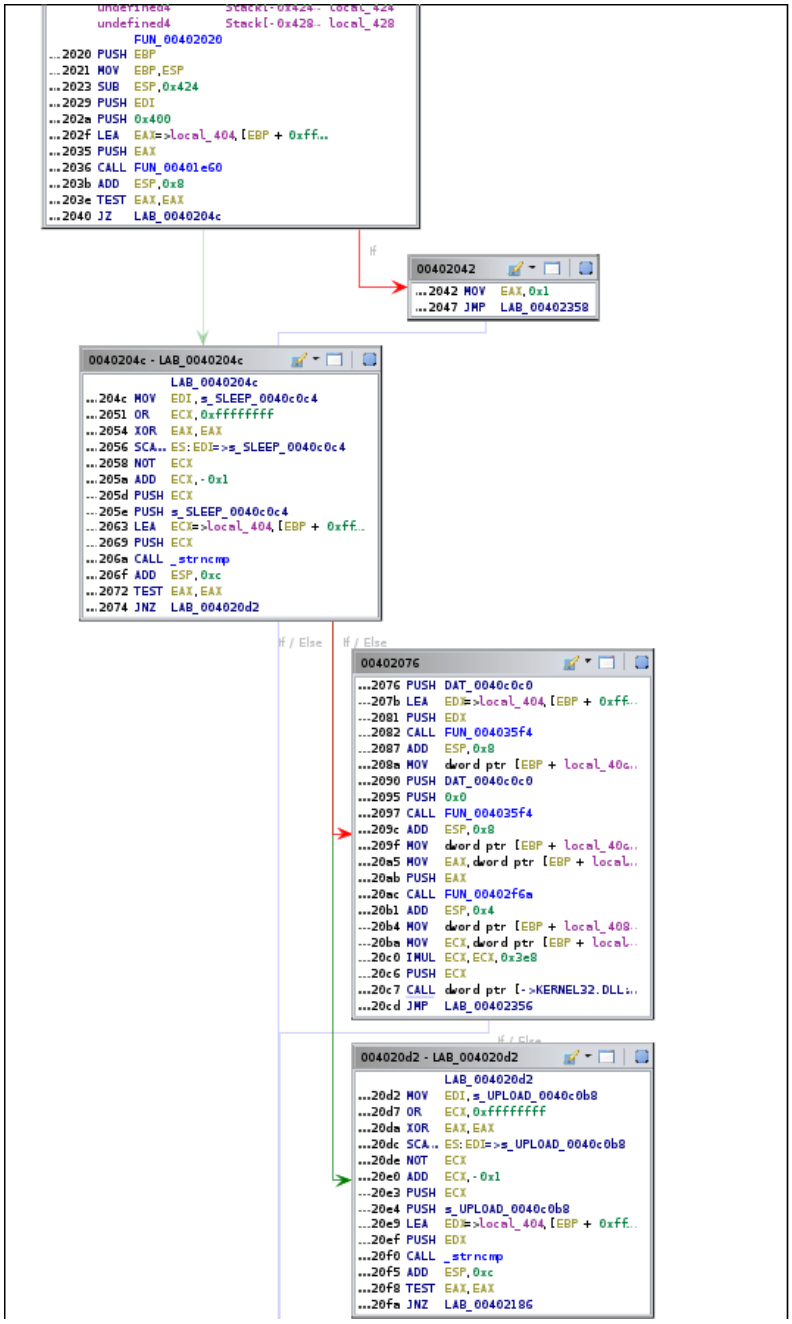
4. Question 4



One possible way of detecting the malware is by checking if any registry values were added. These registry keys are added to create persistance.

We can check for these registry keys as a way of detecting whether a computer has been infected or not

5. Question 5



Taking a look at 0x00402020, we can see that there are multiple different tasks the malware does

(a) Sleep

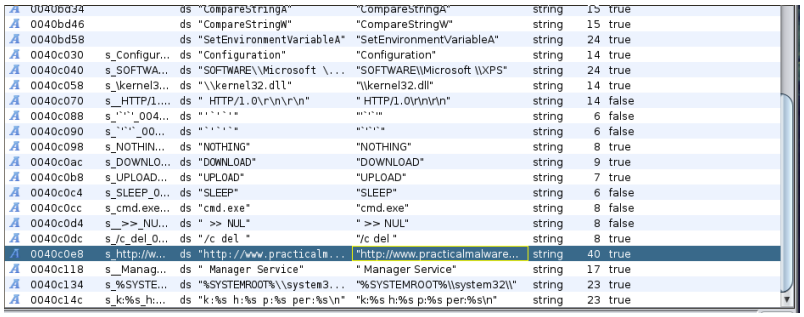
(b) Upload

(c) Download

(d) Execute

(e) Do nothing

6. Question 6



We can see in the strings, there is a website stored in the program, `"http://www.practicalmalwareanalysis.com"`. Using wireshark, we see that the malware is trying to receive commands from the website

1.1.2 Lab 9-2

1. Question 1

004001c8		IMAGE_SECTION_HEADER	".text"	string	6	false
004040cc		ds "runtime error "	"runtime error "	string	15	true
004040e0		ds "TLOSS error\r\n"	"TLOSS error\r\n"	string	14	true
004040f0		ds "SING error\r\n"	"SING error\r\n"	string	13	true
00404100		ds "DOMAIN error\r\n"	"DOMAIN error\r\n"	string	15	true
00404110		ds "R6028\r\n- unable to ...	"R6028\r\n- unable to initializ...	string	37	true
00404138		ds "R6027\r\n- not enough...	"R6027\r\n- not enough spac...	string	53	true
00404170		ds "R6026\r\n- not enough...	"R6026\r\n- not enough spac...	string	53	true
004041a8		ds "R6025\r\n- pure virtu...	"R6025\r\n- pure virtual functi...	string	38	true
004041d0		ds "R6024\r\n- not enough...	"R6024\r\n- not enough spac...	string	53	true
00404208		ds "R6019\r\n- unable to ...	"R6019\r\n- unable to open c...	string	41	true
00404234		ds "R6018\r\n- unexpected...	"R6018\r\n- unexpected heap...	string	33	true
00404258		ds "R6017\r\n- unexpected...	"R6017\r\n- unexpected multi...	string	45	true
00404288		ds "R6016\r\n- not enough...	"R6016\r\n- not enough spac...	string	44	true
004042b4		ds "\r\nabnormal program ...	"\r\nabnormal program termin...	string	33	true
004042d8		ds "R6009\r\n- not enough...	"R6009\r\n- not enough spac...	string	44	true
00404304	s_R6008_-...	ds "R6008\r\n- not enough...	"R6008\r\n- not enough spac...	string	42	true
00404330		ds "R6002\r\n- floating p...	"R6002\r\n- floating point not ...	string	37	true
00404358	s_Microsof...	ds "Microsoft Visual C++ ...	"Microsoft Visual C++ Runtim...	string	37	true
00404384	s_Runtime...	ds "Runtime Error!\n\nPro...	"Runtime Error!\n\nProgram: "	string	26	true
004043a4	s_<progra...	ds "<program name unknown>"	"<program name unknown>"	string	23	true
004043bc	s_GetLast...	ds "GetLastActivePopup"	"GetLastActivePopup"	string	19	true
004043d0	s_GetActiv...	ds "GetActiveWindow"	"GetActiveWindow"	string	16	true
004043e0	s_Messag...	ds "MessageBoxA"	"MessageBoxA"	string	12	true
004043ec	s_user32....	ds "user32.dll"	"user32.dll"	string	11	true
0040451e		ds "WaitForSingleObject"	"WaitForSingleObject"	string	20	true
00404534		ds "CreateProcessA"	"CreateProcessA"	string	15	true
00404546		ds "Sleep"	"Sleep"	string	6	false
0040454e		ds "GetModuleFileNameA"	"GetModuleFileNameA"	string	19	true
00404562		ds "KERNEL32.dll"	"KERNEL32.dll"	string	13	true
00404572		ds "WSASocketA"	"WSASocketA"	string	11	true
0040457e		ds "WS2_32.dll"	"WS2_32.dll"	string	11	true
0040458c		ds "GetCommandLineA"	"GetCommandLineA"	string	16	true
0040459e		ds "GetVersion"	"GetVersion"	string	11	true
004045ac		ds "ExitProcess"	"ExitProcess"	string	12	true
004045ba		ds "TerminateProcess"	"TerminateProcess"	string	17	true
004045ce		ds "GetCurrentProcess"	"GetCurrentProcess"	string	18	true
004045e2		ds "UnhandledExceptionFil...	"UnhandledExceptionFilter"	string	25	true
004045fe		ds "FreeEnvironmentStringsA"	"FreeEnvironmentStringsA"	string	24	true
00404618		ds "FreeEnvironmentStringsW"	"FreeEnvironmentStringsW"	string	24	true
00404632		ds "WideCharToMultiByte"	"WideCharToMultiByte"	string	20	true
00404648		ds "GetEnvironmentStrings"	"GetEnvironmentStrings"	string	22	true
00404660		ds "GetEnvironmentStringsW"	"GetEnvironmentStringsW"	string	23	true

004043e0	s_Messag...	ds "MessageBoxA"	"MessageBoxA"	string	12	true
004043ec	s_user32....	ds "user32.dll"	"user32.dll"	string	11	true
0040451e		ds "WaitForSingleObject"	"WaitForSingleObject"	string	20	true
00404534		ds "CreateProcessA"	"CreateProcessA"	string	15	true
00404546		ds "Sleep"	"Sleep"	string	6	false
0040454e		ds "GetModuleFileNameA"	"GetModuleFileNameA"	string	19	true
00404562		ds "KERNEL32.dll"	"KERNEL32.dll"	string	13	true
00404572		ds "WSASocketA"	"WSASocketA"	string	11	true
0040457e		ds "WS2_32.dll"	"WS2_32.dll"	string	11	true
0040458c		ds "GetCommandLineA"	"GetCommandLineA"	string	16	true
0040459e		ds "GetVersion"	"GetVersion"	string	11	true
004045ac		ds "ExitProcess"	"ExitProcess"	string	12	true
004045ba		ds "TerminateProcess"	"TerminateProcess"	string	17	true
004045ce		ds "GetCurrentProcess"	"GetCurrentProcess"	string	18	true
004045e2		ds "UnhandledExceptionFil...	"UnhandledExceptionFilter"	string	25	true
004045fe		ds "FreeEnvironmentStringsA"	"FreeEnvironmentStringsA"	string	24	true
00404618		ds "FreeEnvironmentStringsW"	"FreeEnvironmentStringsW"	string	24	true
00404632		ds "WideCharToMultiByte"	"WideCharToMultiByte"	string	20	true
00404648		ds "GetEnvironmentStrings"	"GetEnvironmentStrings"	string	22	true
00404660		ds "GetEnvironmentStringsW"	"GetEnvironmentStringsW"	string	23	true
0040467a		ds "SetHandleCount"	"SetHandleCount"	string	15	true
0040468c		ds "GetStdHandle"	"GetStdHandle"	string	13	true
0040469c		ds "GetFileType"	"GetFileType"	string	12	true
004046aa		ds "GetStartupInfoA"	"GetStartupInfoA"	string	16	true
004046bc		ds "HeapDestroy"	"HeapDestroy"	string	12	true
004046ca		ds "HeapCreate"	"HeapCreate"	string	11	true
004046d8		ds "VirtualFree"	"VirtualFree"	string	12	true
004046e6		ds "HeapFree"	"HeapFree"	string	9	true
004046f2		ds "RtlUnwind"	"RtlUnwind"	string	10	true
004046fe		ds "WriteFile"	"WriteFile"	string	10	true
0040470a		ds "HeapAlloc"	"HeapAlloc"	string	10	true
00404716		ds "GetCPInfo"	"GetCPInfo"	string	10	true
00404722		ds "GetACP"	"GetACP"	string	7	false
0040472c		ds "GetOEMCP"	"GetOEMCP"	string	9	true
00404738		ds "VirtualAlloc"	"VirtualAlloc"	string	13	true
00404748		ds "HeapReAlloc"	"HeapReAlloc"	string	12	true
00404756		ds "GetProcAddress"	"GetProcAddress"	string	15	true
00404768		ds "LoadLibraryA"	"LoadLibraryA"	string	13	true
00404778		ds "MultiByteToWideChar"	"MultiByteToWideChar"	string	20	true
0040478e		ds "LCMapStringA"	"LCMapStringA"	string	13	true
0040479e		ds "LCMapStringW"	"LCMapStringW"	string	13	true
004047ae		ds "GetStringTypeA"	"GetStringTypeA"	string	15	true
004047c0		ds "GetStringTypeW"	"GetStringTypeW"	string	15	true

ds "HeapAlloc"

2. Question 2

   The malware terminates immediately

3. Question 3

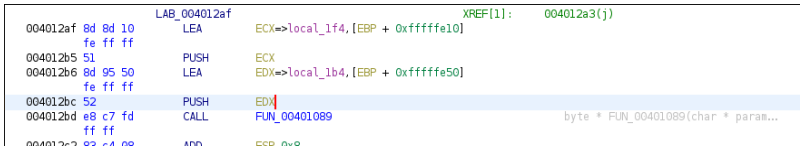
Analyzing the function above, we can see te binary obtains the name of its executable using `GetModuleFileNameA`. It strips this path and then compares the filename to ocl.exe. If these don't match, the malware will terminate.

4. Question 4

```
MOV        byte ptr [EBP + local_1b4],0x31
MOV        byte ptr [EBP + local_1b3],0x71
MOV        byte ptr [EBP + local_1b2],0x61
MOV        byte ptr [EBP + local_1b1],0x7a
MOV        byte ptr [EBP + local_1b0],0x32
MOV        byte ptr [EBP + local_1af],0x77
MOV        byte ptr [EBP + local_1ae],0x73
MOV        byte ptr [EBP + local_1ad],0x78
MOV        byte ptr [EBP + local_1ac],0x33
MOV        byte ptr [EBP + local_1ab],0x65
MOV        byte ptr [EBP + local_1aa],0x64
MOV        byte ptr [EBP + local_1a9],0x63
MOV        byte ptr [EBP + local_1a8],0x0
MOV        byte ptr [EBP + local_1a4],0x6f
MOV        byte ptr [EBP + local_1a3],0x63
MOV        byte ptr [EBP + local_1a2],0x6c
MOV        byte ptr [EBP + local_1a1],0x2e
MOV        byte ptr [EBP + local_1a0],0x65
MOV        byte ptr [EBP + local_19f],0x78
MOV        byte ptr [EBP + local_19e],0x65
```

We see that a string is being formed character by character. The string ends up becoming `"1qaz2wsx3edc"`. This was done to prevent disassemblers from detecting it as a proper string through static analysis.

5. Question 5



We see that a string is being passed. When we use dynamic analysis on the program, we see that the string that was created above was passed into the subroutine. The pointer `0x0012FD90` is also passed in

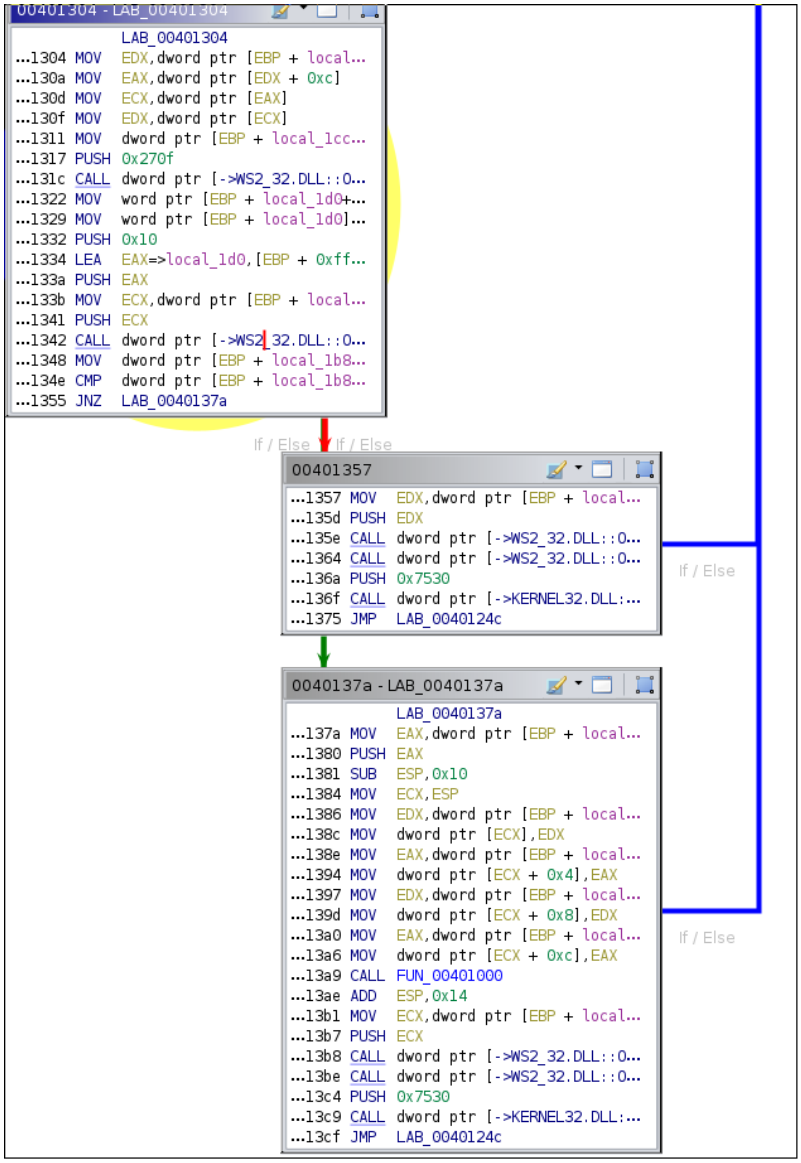
6. Question 6

   `http://www.practicalmalwareanalysis.com`

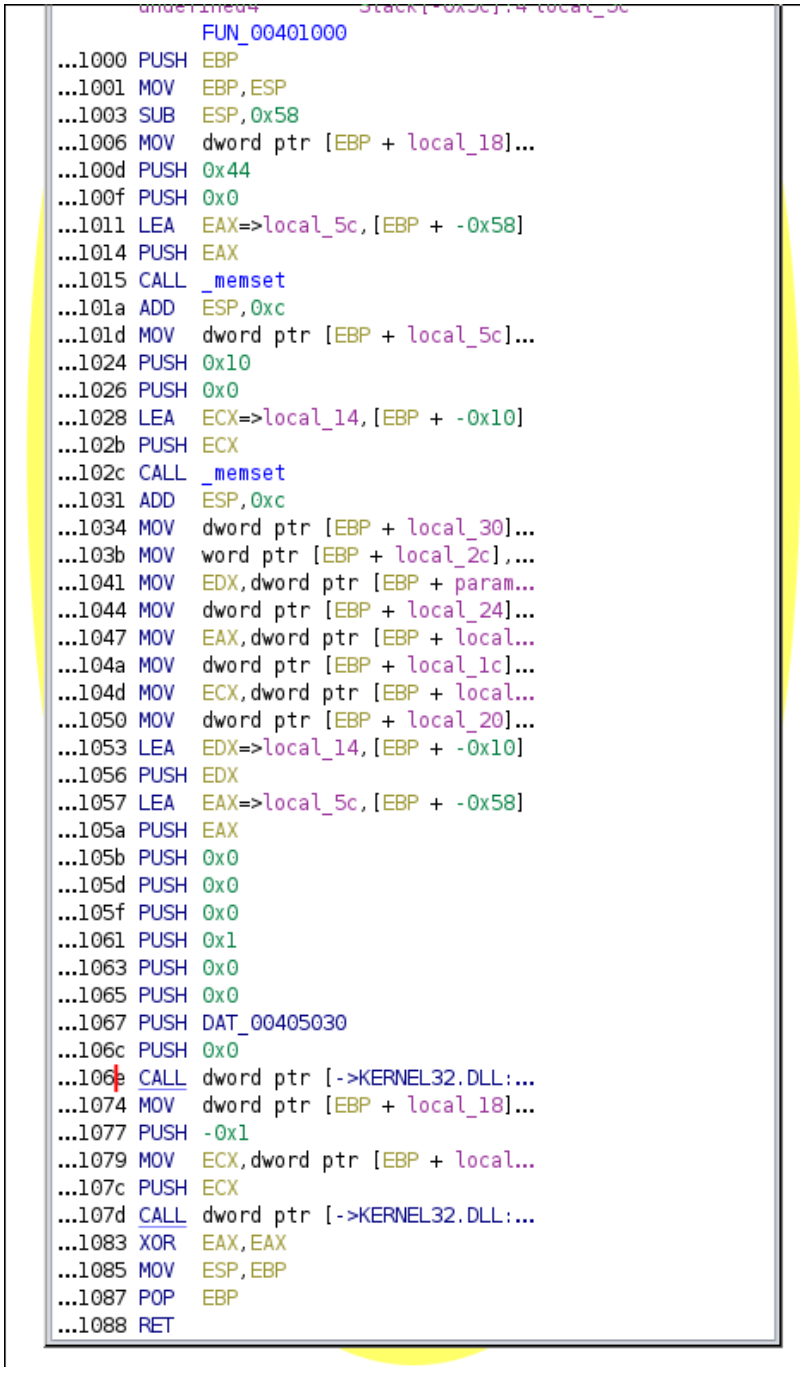
7. Question 7

   The malware uses XOR to obfuscate the domain name

8. Question 8



The first block decodes the comain name and gets the ip by using `gethostbyname`. It then tried to connect to the ip at port 9999. Then the socket is passed into the `0x00401000`.



```
                undefined4          Stack[-0x3c]:4 local_3c
                FUN_00401000
...1000 PUSH  EBP
...1001 MOV   EBP,ESP
...1003 SUB   ESP,0x58
...1006 MOV   dword ptr [EBP + local_18]...
...100d PUSH  0x44
...100f PUSH  0x0
...1011 LEA   EAX=>local_5c,[EBP + -0x58]
...1014 PUSH  EAX
...1015 CALL  _memset
...101a ADD   ESP,0xc
...101d MOV   dword ptr [EBP + local_5c]...
...1024 PUSH  0x10
...1026 PUSH  0x0
...1028 LEA   ECX=>local_14,[EBP + -0x10]
...102b PUSH  ECX
...102c CALL  _memset
...1031 ADD   ESP,0xc
...1034 MOV   dword ptr [EBP + local_30]...
...103b MOV   word ptr [EBP + local_2c],...
...1041 MOV   EDX,dword ptr [EBP + param...
...1044 MOV   dword ptr [EBP + local_24]...
...1047 MOV   EAX,dword ptr [EBP + local...
...104a MOV   dword ptr [EBP + local_1c]...
...104d MOV   ECX,dword ptr [EBP + local...
...1050 MOV   dword ptr [EBP + local_20]...
...1053 LEA   EDX=>local_14,[EBP + -0x10]
...1056 PUSH  EDX
...1057 LEA   EAX=>local_5c,[EBP + -0x58]
...105a PUSH  EAX
...105b PUSH  0x0
...105d PUSH  0x0
...105f PUSH  0x0
...1061 PUSH  0x1
...1063 PUSH  0x0
...1065 PUSH  0x0
...1067 PUSH  DAT_00405030
...106c PUSH  0x0
...106e CALL  dword ptr [->KERNEL32.DLL:...
...1074 MOV   dword ptr [EBP + local_18]...
...1077 PUSH  -0x1
...1079 MOV   ECX,dword ptr [EBP + local...
...107c PUSH  ECX
...107d CALL  dword ptr [->KERNEL32.DLL:...
...1083 XOR   EAX,EAX
...1085 MOV   ESP,EBP
...1087 POP   EBP
...1088 RET
```

The program then redirects all input and output over console is trans-

mitted over the network. This practically creates a reverse shell to receive commands from the server.