

## Contents

<b>1 Homework</b>	<b>1</b>
1.1 HW7 . . . . .	1
1.1.1 Lab 9-1 . . . . .	1
1.1.2 Lab 9-2 . . . . .	5

## 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 function, we see that the passcode is "abcd".

	LAB_00402b1d		XREF[1]:	00402b01(j)
00402b1d 8b 45 08	MOV	EAX, dword ptr [EBP + param_1]		
00402b20 8b 4d 0c	MOV	ECX, dword ptr [EBP + param_2]		
00402b23 8b 54 81 fc	MOV	EDX, dword ptr [ECX + EAX*0x4 + -0x4]		
00402b27 89 55 fc	MOV	dword ptr [EBP + local_8], EDX		
00402b2a 8b 45 fc	MOV	EAX, dword ptr [EBP + local_8]		
00402b2d 50	PUSH	EAX		
00402b2e e8 dd f9	CALL	FUN_00402510		undefined4 FUN_00402510(char * p...
ff ff				
00402b33 83 c4 04	ADD	ESP, 0x4		
00402b36 85 c0	TEST	EAX, EAX		
00402b38 75 05	JNZ	LAB_00402b3f		
00402b3a e8 d1 f8	CALL	FUN_00402410		undefined FUN_00402410(void)
ff ff				

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

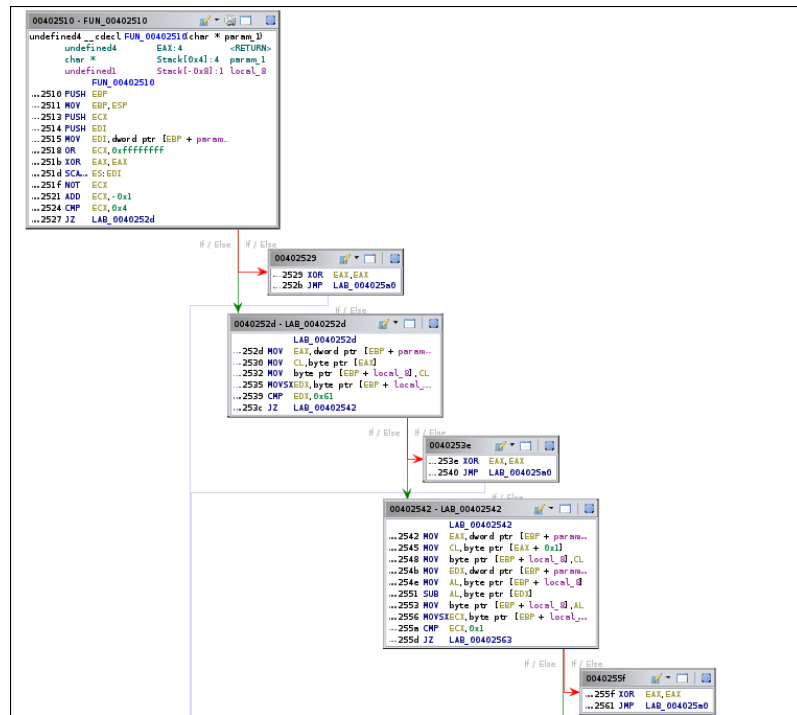
We can install the malware by executing it with the arguments "Lab09-01.exe -in abcd".

##### 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".

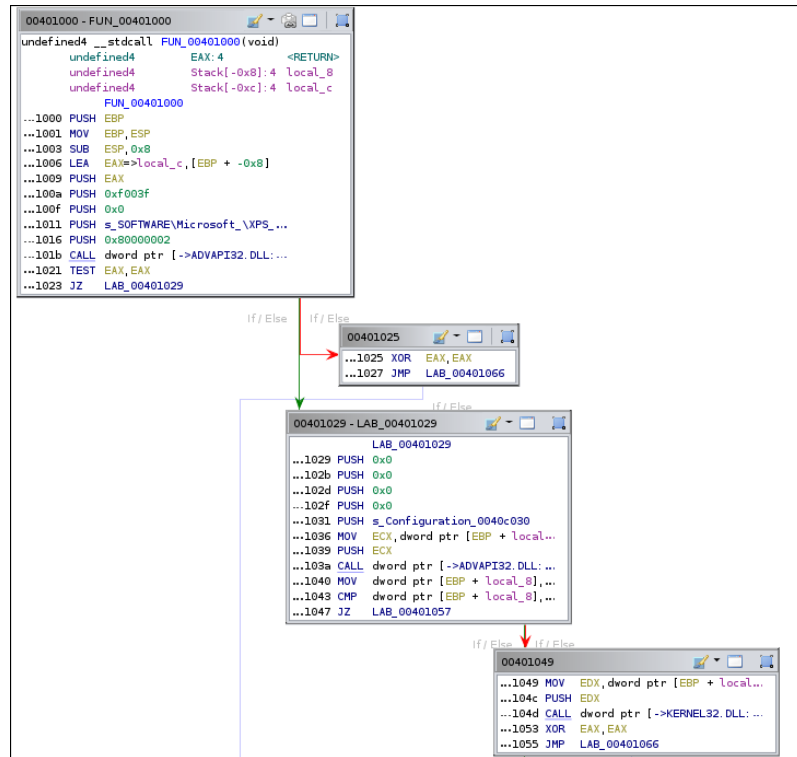
### 3. Question 3

To Patch the program so that it doesnt require a password, we need to patch 0x00402b38 from jnz to jz.

	LAB_00402b1d		XREF[1]: 00402b01(j)
00402b1d 8b 45 08	MOV	EAX,dword ptr [EBP + param_1]	
00402b20 8b 4d 0c	MOV	ECX,dword ptr [EBP + param_2]	
00402b23 8b 54 81 fc	MOV	EDX,dword ptr [ECX + EAX*0x4 + -0x4]	
00402b27 89 55 fc	MOV	dword ptr [EBP + local_8],EDX	
00402b2a 8b 45 fc	MOV	EAX,dword ptr [EBP + local_8]	
00402b2d 50	PUSH	EAX	
00402b2e e8 dd f9	CALL	FUN_00402510	undefined4 FUN_00402510(char * p...
ff ff			
00402b33 83 c4 04	ADD	ESP,0x4	
00402b36 85 c0	TEST	EAX,EAX	
00402b38 74 05	JZ	LAB_00402b3f	
00402b3a e8 d1 f8	CALL	FUN_00402410	undefined FUN_00402410(void)
ff ff			

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

#### 4. Question 4

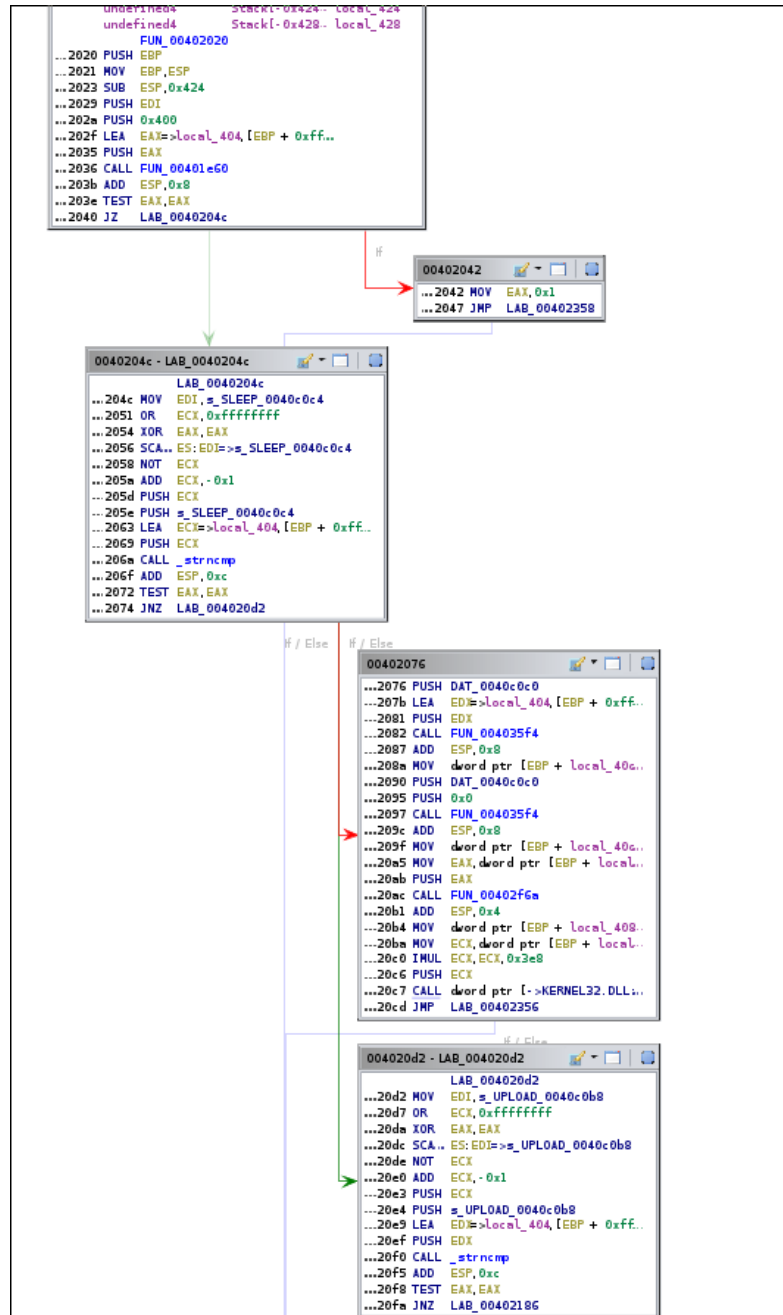


One possible way of detecting the malware is by checking if any registry values were added. These registry keys are added to create persistence.

```
HKLM\SYSTEM\ControlSet001\Services\Lab09-01_patched\Type: 0x00000020
HKLM\SYSTEM\ControlSet001\Services\Lab09-01_patched\Start: 0x00000002
HKLM\SYSTEM\ControlSet001\Services\Lab09-01_patched\ErrorControl: 0x00000001
HKLM\SYSTEM\ControlSet001\Services\Lab09-01_patched\ImagePath: "SYSTEMROOT\system32\Lab09-01_patched.exe"
HKLM\SYSTEM\ControlSet001\Services\Lab09-01_patched\DisplayName: "Lab09-01_patched Manager Service"
HKLM\SYSTEM\ControlSet001\Services\Lab09-01_patched\ObjectName: "LocalSystem"
HKLM\SYSTEM\ControlSet001\Services\Lab09-01_patched\Security\security: 01 00 14 80 90 00 00 00 9C 00 00 00 14 0C
HKLM\SYSTEM\CurrentControlSet\Services\Lab09-01_patched\Type: 0x00000020
HKLM\SYSTEM\CurrentControlSet\Services\Lab09-01_patched\Start: 0x00000002
HKLM\SYSTEM\CurrentControlSet\Services\Lab09-01_patched\ErrorControl: 0x00000001
HKLM\SYSTEM\CurrentControlSet\Services\Lab09-01_patched\ImagePath: "SYSTEMROOT\system32\Lab09-01_patched.exe"
HKLM\SYSTEM\CurrentControlSet\Services\Lab09-01_patched\DisplayName: "Lab09-01_patched Manager Service"
HKLM\SYSTEM\CurrentControlSet\Services\Lab09-01_patched\ObjectName: "LocalSystem"
```

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

0040b054	ds	"CompareStringA"	"CompareStringA"	string	15	true
0040bd46	ds	"CompareStringW"	"CompareStringW"	string	15	true
0040bd58	ds	"SetEnvironmentVariableA"	"SetEnvironmentVariableA"	string	24	true
0040c030	s_Configur...	ds	"Configuration"	string	14	true
0040c040	s_SOFTWA...	ds	"SOFTWARE\\Microsoft \\...	string	24	true
0040c058	s_kernel3...	ds	"\\kernel32.dll"	string	14	true
0040c070	s_HTTP/1...	ds	" HTTP/1.0\\r\\n\\r\\n"	string	14	false
0040c088	s_..._004...	ds	"..."	string	6	false
0040c090	s_..._00...	ds	"..."	string	6	false
0040c098	s_NOTHIN...	ds	"NOTHING"	string	8	true
0040c0ac	s_DOWNLO...	ds	"DOWNLOAD"	string	9	true
0040c0b8	s_UPLOAD...	ds	"UPLOAD"	string	7	true
0040c0c4	s_SLEEP_0...	ds	"SLEEP"	string	6	false
0040c0cc	s_cmd.exe...	ds	"cmd.exe"	string	8	false
0040c0d4	s_>>_NUL...	ds	" >> NUL"	string	8	false
0040c0dc	s_/c del 0...	ds	"/c del "	string	8	true
0040c0e8	s_http://w...	ds	"http://www.practicals..."	string	40	true
0040c118	s_Manag...	ds	"Manager Service"	string	17	true
0040c134	s_%SYSTE...	ds	"%SYSTEMROOT%\\system3...	string	23	true
0040c14c	s_k:%s_h...	ds	"k:%s h:%s p:%s per:%s\\n"	string	23	true

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

A 004001c8	IMAGE_SECTION_HEADER	".text"	string	0	false
A 004040cc	ds "runtime error "	"runtime error "	string	15	true
A 004040e0	ds "TLOSS error\r\n"	"TLOSS error\r\n"	string	14	true
A 004040f0	ds "SING error\r\n"	"SING error\r\n"	string	13	true
A 00404100	ds "DOMAIN error\r\n"	"DOMAIN error\r\n"	string	15	true
A 00404110	ds "R6028\r\n- unable to ...	"R6028\r\n- unable to initializ...	string	37	true
A 00404138	ds "R6027\r\n- not enough...	"R6027\r\n- not enough spac...	string	53	true
A 00404170	ds "R6026\r\n- not enough...	"R6026\r\n- not enough spac...	string	53	true
A 004041a8	ds "R6025\r\n- pure virtu...	"R6025\r\n- pure virtual functi...	string	38	true
A 004041d0	ds "R6024\r\n- not enough...	"R6024\r\n- not enough spac...	string	53	true
A 00404208	ds "R6019\r\n- unable to ...	"R6019\r\n- unable to open c...	string	41	true
A 00404234	ds "R6018\r\n- unexpected...	"R6018\r\n- unexpected heap...	string	33	true
A 00404258	ds "R6017\r\n- unexpected...	"R6017\r\n- unexpected multi...	string	45	true
A 00404288	ds "R6016\r\n- not enough...	"R6016\r\n- not enough spac...	string	44	true
A 004042b4	ds "\r\nabnormal program ...	"\r\nabnormal program termin...	string	33	true
A 004042d8	ds "R6009\r\n- not enough...	"R6009\r\n- not enough spac...	string	44	true
A 00404304	s_R6008_...	ds "R6008\r\n- not enough...	string	42	true
A 00404330	ds "R6002\r\n- floating p...	"R6002\r\n- floating point not ...	string	37	true
A 00404358	s_Microsof...	ds "Microsoft Visual C++ ...	string	37	true
A 00404384	s_Runtime...	ds "Runtime Error!\n\nPro...	string	26	true
A 004043a4	s_<progra...	ds "<program name unknown>"	string	23	true
A 004043bc	s_GetLast...	ds "GetLastActivePopup"	string	19	true
A 004043d0	s_GetActiv...	ds "GetActiveWindow"	string	16	true
A 004043e0	s_Message...	ds "MessageBoxA"	string	12	true
A 004043ec	s_user32...	ds "user32.dll"	string	11	true
A 0040451e	ds "WaitForSingleObject"	"WaitForSingleObject"	string	20	true
A 00404534	ds "CreateProcessA"	"CreateProcessA"	string	15	true
A 00404546	ds "Sleep"	"Sleep"	string	6	false
A 0040454e	ds "GetModuleFileNameA"	"GetModuleFileNameA"	string	19	true
A 00404562	ds "KERNEL32.dll"	"KERNEL32.dll"	string	13	true
A 00404572	ds "WSocketA"	"WSocketA"	string	11	true
A 0040457e	ds "WS2_32.dll"	"WS2_32.dll"	string	11	true
A 0040458c	ds "GetCommandLineA"	"GetCommandLineA"	string	16	true
A 0040459e	ds "GetVersion"	"GetVersion"	string	11	true
A 004045ac	ds "ExitProcess"	"ExitProcess"	string	12	true
A 004045ba	ds "TerminateProcess"	"TerminateProcess"	string	17	true
A 004045ce	ds "GetCurrentProcess"	"GetCurrentProcess"	string	18	true
A 004045e2	ds "UnhandledExceptionFil...	"UnhandledExceptionFilter"	string	25	true
A 004045fe	ds "FreeEnvironmentStringsA"	"FreeEnvironmentStringsA"	string	24	true
A 00404618	ds "FreeEnvironmentStringsW"	"FreeEnvironmentStringsW"	string	24	true
A 00404632	ds "WideCharToMultiByte"	"WideCharToMultiByte"	string	20	true
A 00404648	ds "GetEnvironmentStrings"	"GetEnvironmentStrings"	string	22	true
A 00404660	ds "GetEnvironmentStringsW"	"GetEnvironmentStringsW"	string	23	true

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

2. Question 2

The malware terminates immediately

3. Question 3

The screenshot displays a debugger window with assembly code on the left and a control flow graph on the right. The assembly code, starting at address 00401128, includes instructions such as `PUSH EBP`, `MOV EBP, ESP`, `SUB ESP, 0x304`, and a series of `MOV` instructions for local variables. It concludes with `JZ LAB_0040124c` at address 00401240. The control flow graph on the right shows a jump from address 00401242 to 004013d6, and another jump from address 0040124c to 004013d6. The graph uses red and green nodes to represent different execution paths.

```
00401128  PUSH EBP
00401129  MOV  EBP, ESP
0040112b  SUB  ESP, 0x304
00401131  PUSH ESI
00401132  PUSH EDI
00401133  MOV  byte ptr [EBP + local_1b4], ...
0040113a  MOV  byte ptr [EBP + local_1b3], ...
00401141  MOV  byte ptr [EBP + local_1b2], ...
00401148  MOV  byte ptr [EBP + local_1b1], ...
0040114f  MOV  byte ptr [EBP + local_1b0], ...
00401156  MOV  byte ptr [EBP + local_1af], ...
0040115d  MOV  byte ptr [EBP + local_1ae], ...
00401164  MOV  byte ptr [EBP + local_1ad], ...
0040116b  MOV  byte ptr [EBP + local_1ac], ...
00401172  MOV  byte ptr [EBP + local_1ab], ...
00401179  MOV  byte ptr [EBP + local_1aa], ...
00401180  MOV  byte ptr [EBP + local_1a9], ...
00401187  MOV  byte ptr [EBP + local_1a8], ...
0040118e  MOV  byte ptr [EBP + local_1a7], ...
00401195  MOV  byte ptr [EBP + local_1a6], ...
0040119c  MOV  byte ptr [EBP + local_1a5], ...
004011a3  MOV  byte ptr [EBP + local_1a4], ...
004011aa  MOV  byte ptr [EBP + local_1a3], ...
004011b1  MOV  byte ptr [EBP + local_1a2], ...
004011b8  MOV  byte ptr [EBP + local_1a1], ...
004011bf  MOV  byte ptr [EBP + local_1a0], ...
004011c6  MOV  ECX, 0x8
004011cb  MOV  ESI, DAT_00405034
004011d0  LEA  EDI, >local_1f4, [EBP + 0xff...]
004011d6  MOV  ... ES: EDI, ESI => DAT_00405034
004011d8  MOVSB ES: EDI, ESI => DAT_00405034
004011d9  MOV  dword ptr [EBP + local_1bc], ...
004011e3  MOV  byte ptr [EBP + local_304], ...
004011ea  MOV  ECX, 0x43
004011ef  XOR  EAX, EAX
004011f1  LEA  EDI, >local_303, [EBP + 0xff...]
004011f7  STQ  ... ES: EDI
004011f9  STOSB ES: EDI
004011fa  PUSH 0x10e
004011ff  LEA  EAX, >local_304, [EBP + 0xff...]
00401205  PUSH EAX
00401206  PUSH 0x0
00401208  CALL dword ptr [->KERNEL32.DLL...]
0040120e  PUSH 0x5c
00401210  LEA  ECX, >local_304, [EBP + 0xff...]
00401216  PUSH ECX
00401217  CALL _strrchr
0040121c  ADD  ESP, 0x8
0040121f  MOV  dword ptr [EBP + local_8], ...
00401222  MOV  EDX, dword ptr [EBP + local_...]
00401225  ADD  EDX, 0x1
00401228  MOV  dword ptr [EBP + local_8], ...
0040122b  MOV  EAX, dword ptr [EBP + local_...]
0040122e  PUSH EAX
0040122f  LEA  ECX, >local_1a4, [EBP + 0xff...]
00401235  PUSH ECX
00401236  CALL _strcmp
0040123b  ADD  ESP, 0x8
0040123e  TEST EAX, EAX
00401240  JZ   LAB_0040124c
```

Control Flow Graph details:

- Node 00401242: `MOV EAX, 0x1`, `JMP LAB_004013d6`
- Node 0040124c: (Label)

Analyzing the function above, we can see the 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

004012af	8d 8d 10	LAB_004012af	LEA	ECX=local_1f4,[EBP + 0xfffffe10]	XREF[1]: 004012a3(j)
	fe ff ff				
004012b5	51	PUSH	ECX		
004012b6	8d 95 50	LEA	EDX=local_1b4,[EBP + 0xfffffe50]		
	fe ff ff				
004012bc	52	PUSH	EDX		
004012bd	e9 c7 fd	CALL	FUN_00401089	byte * FUN_00401089(char * param...	
004012c3	92 41 00	ADD	ECX,ECX		

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

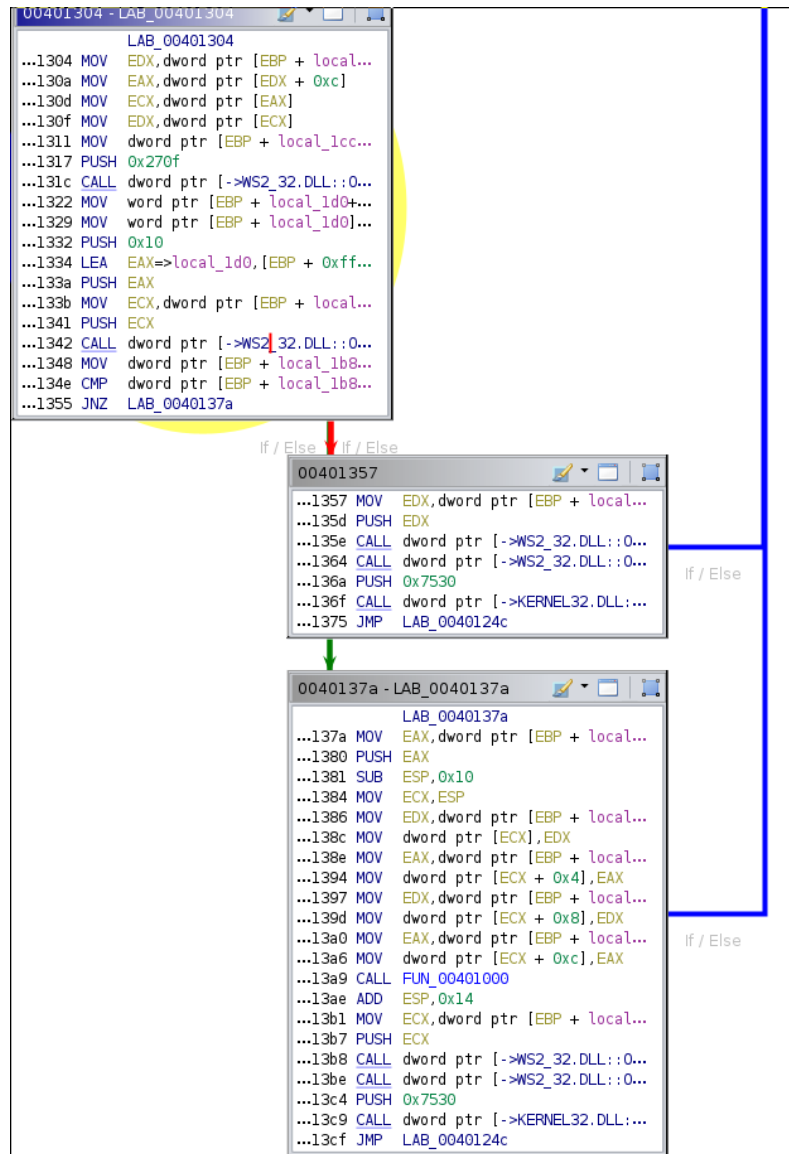
#### 6. Question 6

<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.

```

...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.