Assignment 4

Q-1) Analyze the malware found in the file `Lab09-01.exe` using OllyDbg and Ghidra to answer the following questions. (50pt)

a) How can you get this malware to install itself? (10pt)

The malware can be installed by itself if the `-in` flag option is coupled with the password string. After fixing the program to use the valid command-line arguments,

b) What are the command-line options for this program? What is the password requirement? (10pt)

The password requirement is `abcd` and there are four command-line options: `-in, -re, -c, and -cc`.

Decompile: FUN_00402510 - (Lab09-01.exe)

```
  undefined4 uVar2;
  int iVar3;
  char *pcVar4;

  iVar3 = -1;
  pcVar4 = param_1;
  do {
    if (iVar3 == 0) break;
    iVar3 = iVar3 + -1;
    cVar1 = *pcVar4;
    pcVar4 = pcVar4 + 1;
  } while (cVar1 != '\0');
  if (iVar3 == -6) {
    if (*param_1 == 'a') {
      if ((char)(param_1[1] - *param_1) == '\x01') {
        if (param_1[2] == 'c') {
          if (param_1[3] == 'd') {
            uVar2 = 1;
          }
          else {
            uVar2 = 0;
          }
        }
        else {
          uVar2 = 0;
```

After analyzing the functions on Ghidra, I found this one function at `0x402510` needing the string `abcd` to fully utilize it.

c) How can you use OllyDbg to permanently patch this malware, so that it doesn't require the special command-line password? (10pt)

The password function can be patched to provide any string. I did this by

patching the binary to make it that the password checker will return 1 for all string inputs.

d) What are the host-based indicators of this malware? (5pt)

The main host-based indicator I discovered was a registry key found in the function for `-c` flag. The key, shown below, tells us the software that the host must be using.

Decompile: FUN_00401070 - (Lab09-01.exe)

```
    if (iVar4 == 0) break;
    iVar4 = iVar4 + -1;
    cVar1 = *param_4;
    param_4 = param_4 + 1;
  } while (cVar1 != '\0');
  uStackY52 = 0x4011ae;
  LVar2 = RegCreateKeyExA((HKEY)0x80000002,s_SOFTWARE\Microsoft_\XPS_0040c040,0,(LPSTR)0x0,0,0xf0
                          ,(LPSECURITY_ATTRIBUTES)0x0,&local_1010,(LPDWORD)0x0);
  if (LVar2 == 0) {
    LVar2 = RegSetValueExA(local_1010,s_Configuration_0040c030,0,3,(BYTE *)local_100c,0x1000);
    if (LVar2 == 0) {
      CloseHandle(local_1010);
      uVar3 = 0;
    }
    else {
      CloseHandle(local_1010);
      uVar3 = 1;
    }
  }
  else {
    uVar3 = 1;
  }
  return uVar3;
}
```

e) What are the different actions this malware can be instructed to take via the network? (10pt)

The five different actions that can be taken are: `SLEEP, UPLOAD, CMD, and NOTHING`.

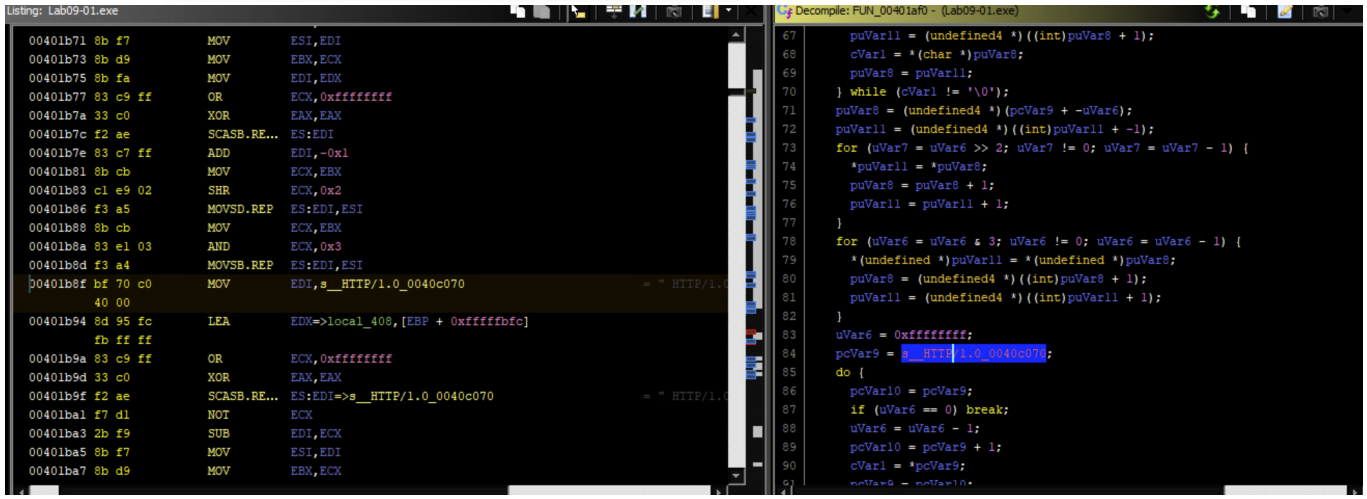
```
  } while (cVar1 != '\0');
  iVar3 = _strncmp((char *)local_404,s_SLEEP_0040c0c4,~uVar8 - 1);
  if (iVar3 == 0) {
    FUN_004035f4((byte *)local_404,&DAT_0040c0c0);
    pbVar4 = (byte *)FUN_004035f4((byte *)0x0,&DAT_0040c0c0);
    iVar3 = FUN_00402f6a(this,pbVar4);
    Sleep(iVar3 * 1000);
  }
  else {
    uVar8 = 0xffffffff;
    pcVar9 = s_UPLOAD_0040c0b8;
    do {
      if (uVar8 == 0) break;
      uVar8 = uVar8 - 1;
      cVar1 = *pcVar9;
      pcVar9 = pcVar9 + 1;
    } while (cVar1 != '\0');
    iVar3 = _strncmp((char *)local_404,s_UPLOAD_0040c0b8,~uVar8 - 1);
    if (iVar3 == 0) {
```

```
      else {
        uVar8 = 0xffffffff;
        pcVar9 = s_NOTHING_0040c098;
        do {
          if (uVar8 == 0) break;
          uVar8 = uVar8 - 1;
          cVar1 = *pcVar9;
          pcVar9 = pcVar9 + 1;
        } while (cVar1 != '\0');
        _strncmp((char *)local_404,s_NOTHING_0040c098,~uVar8 - 1);
      }
    }
  }
}
byte * param_1, byte * param_2)
```

f) Are there any useful network-based signatures for this malware? (5pt)

Yes HTTP/1.0 Requests are used shown by the code below. Using HTTP/1.0 is almost always used for outbound connection. There is also

a link to `www.practicalmalwareanalysis`.



Q-2) Analyze the malware found in the file `Lab09-02.exe` using OllyDbg to answer the following questions. (50pt)

a) What strings do you see statically in the binary? (5pt)

The strings you statically see in the binary are the `cmd` string along with strings for imports.

b) What happens when you run this binary? (5pt)

It seems to not do anything and terminate.

c) How can you get this sample to run its malicious payload? (10pt)

If the file is renamed `ocl.exe`, running it will run the malicious payload.

d) What is happening at 0x00401133? (10 pt)

A string is being built one character at a time with the `mov` function. The

string `1qaz2wsx3edc` is formed.

```
00401132  . 57              PUSH EDI
00401133  . C685 50FEFFFF   MOV BYTE PTR SS:[EBP-1B0],31
0040113A  . C685 51FEFFFF   MOV BYTE PTR SS:[EBP-1AF],71
00401141  . C685 52FEFFFF   MOV BYTE PTR SS:[EBP-1AE],61
00401148  . C685 53FEFFFF   MOV BYTE PTR SS:[EBP-1AD],7A
0040114F  . C685 54FEFFFF   MOV BYTE PTR SS:[EBP-1AC],32
00401156  . C685 55FEFFFF   MOV BYTE PTR SS:[EBP-1AB],77
0040115D  . C685 56FEFFFF   MOV BYTE PTR SS:[EBP-1AA],73
00401164  . C685 57FEFFFF   MOV BYTE PTR SS:[EBP-1A9],78
0040116B  . C685 58FEFFFF   MOV BYTE PTR SS:[EBP-1A8],33
00401172  . C685 59FEFFFF   MOV BYTE PTR SS:[EBP-1A7],65
00401179  . C685 5AFEFFFF   MOV BYTE PTR SS:[EBP-1A6],64
00401180  . C685 5BFEFFFF   MOV BYTE PTR SS:[EBP-1A5],63
00401187  . C685 5CFEFFFF   MOV BYTE PTR SS:[EBP-1A4],0
0040118E  . C685 60FEFFFF   MOV BYTE PTR SS:[EBP-1A0],6F
00401195  . C685 61FEFFFF   MOV BYTE PTR SS:[EBP-19F],63
0040119C  . C685 62FEFFFF   MOV BYTE PTR SS:[EBP-19E],6C
004011A3  . C685 63FEFFFF   MOV BYTE PTR SS:[EBP-19D],2E
004011AA  . C685 64FEFFFF   MOV BYTE PTR SS:[EBP-19C],65
004011B1  . C685 65FEFFFF   MOV BYTE PTR SS:[EBP-19B],78
004011B8  . C685 66FEFFFF   MOV BYTE PTR SS:[EBP-19A],65
004011BF  . C685 67FEFFFF   MOV BYTE PTR SS:[EBP-199],0
004011C6  . B9 08000000     MOV ECX,8
004011CB  . BE 34504000     MOV ESI,Lab09-02.00405034
004011D0  . 8DBD 10FEFFFF   LEA EDI,DWORD PTR SS:[EBP-1F0]
004011D6  . F3:A5           REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
004011D8  . A4              MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
004011D9  . C785 48FEFFFF   MOV DWORD PTR SS:[EBP-1B8],0
004011E3  . C685 00FDFFFF   MOV BYTE PTR SS:[EBP-300],0
004011EA  . B9 43000000     MOV ECX,43
004011EF  . 33C0            XOR EAX,EAX
004011F1  . 8DBD 01FDFFFF   LEA EDI,DWORD PTR SS:[EBP-2FF]
004011F7  . F3:AB           REP STOS DWORD PTR ES:[EDI]
004011F9  . AA              STOS BYTE PTR ES:[EDI]
004011FA  . 68 0E010000     PUSH 10E                          BufSize = 10E (270.)
004011FF  . 8D85 00FDFFFF   LEA EAX,DWORD PTR SS:[EBP-300]
00401205  . 50              PUSH EAX                          PathBuffer
00401206  . 6A 00           PUSH 0                            hModule = NULL
00401208  . FF15 0C404000   CALL DWORD PTR DS:[<&KERNEL32.GetModuleFileNameA>]  GetModuleFileNameA
0040120E  . 6A 5C           PUSH 5C
```

e) What arguments are being passed to subroutine 0x00401089? (5pt)

The arguments being passed to the subroutine are the string `1qaz2wsx3edc` as well as an unknown pointer.

f) What domain name does this malware use? (5pt)

The domain name for this malware is `practicalmalwareanalysis.com`.

```
0040110C  . 33CA            XOR ECX,EDX
0040110E  . 8B85 F8FEFFFF   MOV EAX,DWORD PTR SS:[EBP-108]
00401114  . 888C05 00FFFFF  MOV BYTE PTR SS:[EBP+EAX-100],CL
0040111B  .^EB B7           JMP SHORT ccl.004010D4
00401110  > 8D85 00FFFFFF   LEA EAX,DWORD PTR SS:[EBP-100]
00401123  . 5F              POP EDI
00401124  . 8BE5            MOV ESP,EBP
00401126  . 5D              POP EBP
00401127  . C3              RETN
00401128  $ 55              PUSH EBP
00401129  . 8BEC            MOV EBP,ESP
Stack address=0012FB64, (ASCII "www.practicalmalwareanalysis.com")
EAX=0000001F
Jump from 004010EA

Address   Hex dump                  ASCII
0012FB64  77 77 77 2E 70 72 61 63   www.prac
0012FB6C  74 69 63 61 6C 6D 61 6C   ticalmal
0012FB74  77 61 72 65 61 6E 61 6C   wareanal
0012FB7C  79 73 69 73 2E 63 6F 6D   ysis.com
0012FB84  00 00 00 00 00 00 00 00   ........
0012FB8C  00 00 00 00 00 00 00 00   ........
```

g) What encoding routine is being used to obfuscate the domain name? (10pt)

XOR is being used to obfuscate the domain name