Alexander Davenport 40184005

Name: Alexander Davenport
Student Number: 40184005

Q1

(i)

Figure 1a shows a function that looks to gain information about the systems screen resolution using Windows API ‘GetSystemMetrics’. The is most likely to find out whether or not the program is running inside a virtual machine to avoid detection.
The function declares two local variables; ‘var_8’ and ‘var_4’. It calls Windows API ‘GetSystemMetrics’ that deals with the sizes of various objects such as screen, cursors, icons etc. If the function succeeds it returns a value of one and if not the value zero. In the function we can see the API called twice. The first instance moves a value from EAX into ‘[ebp+var_4]’. The API is called again moving the new value in EAX to second variable ‘[ebp+var_8]’. The variables are then pushed onto the stack.
A print function is then performed; “Screen res is: %dx%d\n”. This is our first indication that the program is doing something with the screen resolution values likely stored in ‘var_4’ and ‘var_8 that are screen resolution dimensions.
Next the function caries out a compare instruction; ‘cmp [ebp+var_4], 400h’. This compares the first variable against the hex value of ‘1024’. A jump instruction ‘jnz’ is used to jump if the zero flag is not set. If ‘var_4’ is equal to ‘1024’ the zero flag is set to one and the program continues, if not it will jump to ‘loc_401168’ where it then unconditional jumps to the end of the function.
After the first successful compare or IF statement another compare instruction is carried out against the second variable ‘ebp+var_8’. It compares this variable to the hex value of ‘768’. If the value matches, a conditional jump is performed using ‘jnz’ to the same address as the last IF statement to ‘loc_401168’ where it moves zero into the EAX register and unconditional jumps to the end of the function. If the value is not matched the zero flag is not set and the function continues to the next instruction block that moves the value of ono into register EAX before unconditional jump to the end of the program.
The values either zero or one moved into the EAX register is a significant indication that the program will return zero if the program thinks it is running in a virtual environment or one if it is not and is present on a live system.

Alexander Davenport 40184005

Name: Alexander Davenport
Student Number: 40184005

(ii)

The major code constructs used is two IF statements. This is indicted by two branches after a compare and jump instruction. The first is shown in the code below where it compares hex value '400h' or '1024' in decimal to a value in location '[ebp+var_4]' and jumps if the zero flag is set '1' or continues if the zero flag is not set '0'.

```
cmp     [ebp+var_4], 400h
jnz     short loc_401168
```

The second is shown in the code below where it compares hex value '300h' or '768' in decimal to a value in location '[ebp+var_8]' and jumps if the zero flag is set '1' or continues if the zero flag is not set '0'.

```
cmp     [ebp+var_8], 300h
jnz     short loc_401168
```

Alexander Davenport 40184005

Name: Alexander Davenport
Student Number: 40184005

Q2

(i)

This function appears to look for an internet connection in order to download a file, read information from that file before carrying out some sort of command and sleeping in the background of the infected system.
The program begins with initialising nine local variables. These variables are the first indication that the function is doing something with the internet and a file due to; ‘hFile’, and ‘hInternet’.
The second indication is that it uses the value passed from the function in Firgure 1 in its first IF Statement. It calls subroutine ‘401125’ and compares that value passed into register EAX to one. Using ‘jnz’ if it matches the function continues and if not, it jumps to ‘loc_4012AA’ that is the end of Figure 2 function. i.e. If the program thinks it is not running in a virtual environment continue or else jump out and end the function.
If the program continues it initialises the use of the Windows API ‘WSAStartup’. This is responsible for running most network and internet applications and allows the program to specify the version of the Windows Sockets required. This is shown by a comment ‘wVersionRequested’ before calling ‘WSAStartup’.
The API ‘InternetOpenA’ is used that initialises the use of the WinINet library. The malware disguises itself as ‘Mozilla/4.0’ to initiate these API’s. All other parameters passed to from the function are zero. A value is retunred into register EAX from this API. The value in EAX is moved into variable ‘[ebp+hInterent]’. An IF statement is then used to compare that variable to zero. If the value is zero and the zero flag is set and therefore unsuccessful it jumps to ‘loc_4011DF’ where it closes the interent handle using API ‘InternetCloseHandle’ and finally exits to ‘loc_4012AA’ at the end of the function.
If the value is not zero the fuction will contine onto the nexr code block that moves zero into variable [ebp+Buffer_400h] before jumping to ‘loc_401288’. An IF statement is carried out here comparing ‘[ebp+Buffer+59h]’ to four. If the value does not match and the zero flag is not set 0, the function will close the connection using API ‘InternetCloseHandle’ before jumping to the end of the function.
If variable is not matched to the value four, the program will jump to ‘loc_4011F0’ where it uses ‘InternetOpenUrl’ to access website ‘http://www.malwarez.com/x.script’ and download and read a file using ‘InternetReadFile’. If the file can be read it will continue to a switch statement construct that will compare multiple values to the value found in the file. There is a total of five comparisons. The value from the file is moved into register eax; ‘movsx eax, byte ptr [ebp+Buffer]. The first comparison to this variable, is ‘3Ch’ followed by ‘21h, ‘64h’, ‘6Fh’ and ‘63h’. Note that the variable [ebp+Buffer] is incremented by one each time it does not find a match through the switch. This could be a form of encryption that allows one command sent that can be interpreted by the malware to carry out a multitude of functions. After the switch statement the Windows API ‘sleep’ is used that suggests that this function is carrying out some sort of command-and-control operating in stealth. The function loops after ‘3600000’ milliseconds to ‘loc_401288’. The major code constructs used by this function are IF statements, a switch statement and a for loop.

Alexander Davenport 40184005

Name: Alexander Davenport
Student Number: 40184005

(ii)
The overall purpose of the malicious function is to download a file located at; ‘http://www.malwarez.com/x.script’. If successful, the function will read the file and compare the command to a series of switch statement arguments that most likely carry out different malicious functions. This could be a command-and-control operation used by the malware author to send information to infected hosts in stealth. If the program is successful it will sleep for ‘3600000’ milliseconds before using a for loop and looping back to ‘loc_401288’ to possibly carryout more malicious functions from the malware author.
The major code constructs used by this function are IF statements, a switch statement and a for loop.