**Basic answers to questions asked in brief.**

1. The purpose of the file is to download a file from a file hosting site and then execute that file on the target system. It also includes deleting a specific file, “Pamela\_Payroll.ppt”, presumably this is an attempt to delete private information but the presence of this file on such systems may be a violation in its own.
2. The main() function of this file is called tmainCRTStartup(), the aim of structure of this function is first in gathering information about the target computer such as the operating system version using “GetVersionExA”. It then checks the managed apps, it initialises things such as: the heap, the multithreaded C runtime library, Run time checks, the I/O system/ It the retrieves the command line and environment strings, it also uses \_\_setargv to parse command line arguments. It then initialises the environment and the C runtime before calling “FUN\_00401170” which is the function that makes a call to an ftp website and downloads a file before executing it. It also deletes a file on the target computer (maybe).
3. Not too sure yet, smth about str to char maybe, although it could be referring to the functions that are called “FUN\_(random numbers)”. This could also refer to FUN\_00401000 which is responsible for forming the inputs to the main function, most likely this option.

To exand on FUN\_00401000 this is a function that takes in 2 values and uses those to allocate values to an array, this array is then used to form “words/strings” that are used in commands later in the program. It is formed using the various predefined strings and data within the program, such as the alphabet both capitalised and not and various numbers/dates and times etc…

1. A different function could be FUN\_0040109 which is responsible for creating and executing the command line argument which sends an ftp request to a predetermined IP to download a file.

A different function could also be \_\_security\_init\_cookie, this is responsible for creating buffer runtime protection for the program being run. It works through putting the cookie on top of the stack and upon exit the value on the stack is compared to the global cookie, a difference between the two implies that an overflow has occured. This is normally called by default, however when using an “entry” point this must be called manually.

1. They bypass most checks here through using windows api like requests to make the antivirus think the requests are normal. The virus also deletes a file to more thorough logs and checks on files that are being deleted are probably in order. Also the fact that a payroll document is on a computer that I think is personal is very concerning as it goes against what was mentioned in the brief.

Explanation of file deletion:

* Using LPCWSTR which is a 32 bit pointer to a constant string of 16 bit Unicode characters.
* &lpFileName\_0040bbc0 is the only reference we have to a file name, lets break this down.
* The command used too do this operation is DeleteFileW(), this takes the arguments as follows: bool DeleteFileW( [in] LPCWSTR lpFileName);
* The actual calling of this function is as follows: DeleteFileW((LPCWSTR)&lpFileName\_0040bbc0);
* The “string” we have is: 0040bbc0, the translation of this is pointing towards the file location that follows: “C:\Users\Petrov\Desktop\Payroll\_Pamella5513.ppt”.
* How does this translate into the location?
* Active memory inspection, running the exe file whilst having a break point set near the end means all the obfuscated variables have been compiled and are stored in active memory. I simply found the memory addresses of the variables in this text file (memory locations of decrypted strings.txt) then inspecting the active memory at these addresses leads to discovering the plaintext variable values.
* These deobfscuted variables also reveal server information confirming stuff found in wireshark analysis.

Walking through the internet access part(fpt files):

* Starts through attempting an internet connection, only continues if connection is valid.
* This is the next interesting line that is called after establishing the connection: local\_8 = InternetConnectW(local\_20,&DAT\_0040b9c0,0x15,&DAT\_0040b3c0,&DAT\_0040bfc0,1,0x8000000,0 This is establishing the windows api function to connect to the internet to a specific address. The details of this are stored inside various dat locations. (add windows documentation for this function later)
* This variable is then called in this line: FtpGetFileW(local\_8,&lpCommandLine\_0040c1c0,&lpCommandLine\_0040c1c0,0,0x20,2,0);  
  (add documentation for this windows function as well)
* The command line label here points towards specfici memory addresses, the value of the CommandLine label once fully compiled is commandline: 0040C1C0 = 20944.exe. Doesn’t really make sense although the extra data at the end may add to this functionality.
* However knowing what this request looks like in theory makes understanding this code much easier.
* The request looks like this (vaguely in theory), ftp request too 38.111.196.111, whether it is requesting a specific file called 20944.exe, this is because the same label address is in the argument placeholders for the requested file name and the new filename that will be created upon successful completion of the request.
* It then closes the internet connections it initiated.
* The next thing it does is create a new process using the filename aka (20944.exe) as on of the parameters which is the file we just downloaded.
* It then calls WaitForSingleObject(local\_1c.hProcess,0xffffffff); which is calling the process we just created, this is an attempt to execute the .exe we just downloaded.
* After this the function exits

Its worth noting that the deletion of the specific files happens before the downloading and execution of the .exe that is downloaded.

This in a way is a sort of staging malware for a potentially for harmful process.

The deobfuscation functions in this program make use of a common obfuscation technique called string stacks.

The function is called 7 times.

I currently have 5 labels that I have values for so each function creating a label is not a valid approach.

I think I can perform a program to deal with this and figure out what they are doing.

!!!!!!!Unicorn library!!!!!!!

Create python script that takes the machine code as arguments (or the hex values for them)

And shows the values that are created by the weird maths lmao.

In regards the content of the main function its actually FUN\_00401170, this is because the contents of tmainCRTentry is the default code for any C program in order to enter execution.

Explaining each call of FUN\_00401000()

Each one translates too a label that has data, some of them just have 2 values, for example filename has 2 values.