IT&C Security Master Program
The Bucharest University of Economic Studies
Department of Economic Informatics and Cybernetics
Calea Dorobanţi, 15-17, Sector 1, Bucharest, postal code 010552
Email: ism@ase.ro, Website: http://www.ism.ase.ro

Antivirus and Virus Technologies Course

Catalin Boja & Cristian Toma

Implementation requirements (all mandatory)

- **use only instructions for the x8086 processors on 16 bits**
- **implement at least 1 new procedure for each team member (the procedure must receive at least 1 input arguments (pointers or values) on stack and must return a result by the stack or by a register – NO global variables**
- **no compiler errors with TASM or TLINK**

Important notice. If the solution does not contain at least 1 new procedure for each team member (with arguments sent by stack and return by stack or register), you lose 5 points out of 10. For the other 2 points, failing to comply will invalidate your solution (you get 0 points).

The project done in teams will pe presented. For the individual projects, the author will submit also a 15 minute video registration (screen capture and audio) of the demo and the key points of the solution. The video can be shared from a Drive or it can be uploaded on a private YouTube channel. The video must be available during the project evaluation week.

IT&C Security Master Program
The Bucharest University of Economic Studies
Department of Economic Informatics and Cybernetics
Calea Dorobanţi, 15-17, Sector 1, Bucharest, postal code 010552
Email: ism@ase.ro, Website: http://www.ism.ase.ro

Option 1 – Individual assignments

1. CSpawn - Exercise 3, page 48 in Giant Black Book of Computer Viruses (in Resources folder). Rewrite the INFECT_FILE routine as a procedure to give the host a random name and make it a hidden file.....
2. CSpawn - Exercise 4, page 48 in Giant Black Book of Computer Viruses (in Resources folder). Add a procedure to CSpawn which will demand a password before executing the host and will exit without ..... The password will be different for each infected file and will be generated based on the host name (you can choose the pattern)
3. Justin - Exercise 1, page 67 in Giant Black Book of Computer Viruses (in Resources folder). Modify Justin to use a buffer of only 256 bytes to infect a file. To move .....
4. Sequin - Exercise 5, page 97 in Giant Black Book of Computer Viruses (in Resources folder). A virus can simply load itself into memory just below the 640K .....
5. Implement, using at least 1 procedure, the 32-bit register (check the Crypto course) LFSR. The initial seed should be based on residual data from RAM (choose at least 2 sources and XOR them). The LFSR must be able to generate a sequence of 100 bits and print them on the screen
6. Using any of the given viruses, create a simple ransomware virus that will encrypt (with XOR) the hosts and will store the key in memory

Option 2 – Teams of 2 persons

1. CSpawn - Exercise 5, page 48 in Giant Black Book of Computer Viruses (in Resources folder). Add routines to encrypt both the password and the host name in all copies of the virus which are written to disk.....
2. Timid II - Exercise 4, page 86 in Giant Black Book of Computer Viruses (in Resources folder). There is no reason a virus must put itself all at the beginning or at the end .....
3. Sequin - Exercise 7, page 98 in Giant Black Book of Computer Viruses (in Resources folder). A virus could hide in some of the unused RAM between 640K and 1 MB. Develop a strategy ....
4. Using techniques presented in the Polymorphic Viruses chapter (page 425 in Giant Black Book of Computer Viruses (in Resources folder) change any of the presented viruses (MINI, Justin, Timid, Sequin, etc) into a polymorphic one. At each infection the virus should generate a random number of dummy instructions (binary value) and should encrypt some parts of it (the key can be hardcoded)
5. Implement the Playfair cipher in assembly. The solution should receive from command line the password, the name of the input plaintext file and the name of the output cipher text
6. Implement the Vigenere cipher in assembly. The solution should receive from command line the password, the name of the input plaintext file and the name of the output cipher text

7. Implement, using at least 1 procedure, the A5 (check the Crypto course) LFSR. The initial seed should be received as command line arguments for the asm application. The LFSR must be able to generate a sequence of a desired number of bits (that number is also received as command line arguments)
8. Implement a Base64 encoder and decoder. The solution should receive from command line the name of the input plaintext file and the name of the output encoded text

Option 3 – Teams up to 3 persons

Implement a solution that will provide an integrity checker for .com, .exe and .dll files on your computer. You can used code snippets from course examples or from Giant Black Book of Computer Viruses (in Resources folder).

The solution must build a binary/text file that contains

- the full path of the files with the required extensions
- their SHA1 or MD5 hash value in hexadecimal string representation

The application should receive from command line the path to the drive or folder that will be analyzed and checked. For that session, the app will produce the required file. The solution will recursively check for files in all the subfolder of the given path.

If the app receives, besides the path, the file produced by a previous session, it will check if any of the monitored files has been changed. The app will cross check the hash from the file with the one computed from the same file. The app will alert the user by printing a message.

Hints

- the given antivirus example contains multiple routines that can be used to manage input reading
- you have examples from viruses and the antivirus that can be used to recursively check the folder structure
- to get the hash function implementation you can start by reverse engineering the machine code generated from a C implementation (you have C macro based implementations for all SHA1 and MD5 in the Crypto course resources).
- you can translate the C source code implementation for SHA1 or MD5 (you have them in Crypto course resources)