Challenge: Trapped in Time

Objective

The goal of this challenge is to login to **SUN's Domain FTP Server** (on standard **port 21** – other network details will be provided at the beginning of the challenge) using the following credentials:

Username: **disintuitive**
Password: **disintuitiveXXXXY**

XXXXY in the password is a TimeLock code as defined by our TimeLock algorithm...almost. There is an extra character at the end that is simply the middle character of the final hash (i.e., hash length / 2). **Keep integer division in mind.** Of course, the password to the **disintuitive** account changes every 60 seconds! Joy, joy!

Once logged in, you will need to fetch a few files that can somehow be further inspected to reveal one or more messages cleverly embedded within. You are provided no other details, other than that an exorcism may need to be performed, followed by closely inspecting bits and bytes.

Important notes:

- The epoch time for the TimeLock algorithm will be provided at the beginning of the challenge;
- The current time of the machine implementing the TimeLock algorithm for the **disintuitive** user account can be obtained by making a simple socket connection to it on port **54321** (e.g., using netcat, telnet, etc).

Hints:

- Once on the FTP server, the following FTP commands may be useful: `passive`, `binary`, `prompt`, `mget *`.
- Once you determine what to do with the files on the FTP server (and get a result), there are actually **two** hidden files in the result (one may be somewhat...obvious; the other, not so much).
- For the first hidden file, the offset is a power of 2 (and happens to start with that digit *too*), and the interval is a power of 2 (note that the **byte** method may work quite well).
- For the second hidden file, the offset is a power of 2, plus 1, and the interval is a power of 2 that is less than or equal to 4 (note that the **bit** method may work quite well).
- The following bytes most likely "signal" something: 0x00, 0xFF, 0x00, 0x00, 0xFF, 0x00.
- What's a **JFIF**? And is it perhaps involved with the **byte** method?
- It's possible that a **third** hidden file exists inside of one of the two hidden files. Egads! It may be pretty obvious, though. Counting is your friend.

When you think that you have successfully completed the challenge, **send a secret passphrase to your team text channel and @ prof.**

The deliverable for a challenge is a compressed folder submitted on Moodle. The folder must contain following:

(1) A document that provides a ***thorough*** summary of your team's completion of the challenge; include any relevant Steg settings (e.g., -B -o1024 -i8 -wsome_file.bmp) for all stegged documents; include screenshots as appropriate; clearly specify the final **secret passphrase**; note that **each** team member should contribute their own paragraph to this document that describes their role in the challenge;
(2) The "life alert bracelet" image (with its appropriate image type extension);
(3) The "cyber" image (with its appropriate image type extension);
(4) The "Constitution" text (with its appropriate file type extension);
(5) Any modified Python code (e.g., your TimeLock program, XOR program, Steg program); and
(6) Any other scripts or programs that you created to assist you (if applicable).

You will be graded on how well:

(1) You implement TimeLock algorithm to access FTP server and obtain the initial documents (25%);
(2) You decrypt the initial documents to obtain the "life alert bracelet" image (10%);
(3) You determine the Steg details for obtaining the "cyber" image (15%);
(4) You determine the Steg details for obtaining the "Constitution" text (15%);
(5) You determine the Steg details for obtaining the final secret passphrase (15%);
(6) Your document ***thoroughly*** details your team's path to complete the challenge, how well **each** of the members of your team documents their **role** in the challenge, and how accurately you specify the final secret passphrase (20%);

GOOD LUCK!