Homework Assignment 4
Information Security - Theory Vs. Reality
0368-4474
Submit by: 28th of July at 23:59

1 Submission Instructions

The HW assignments will include writing code and answering written questions. The submission will include the requested code file and a PDF file with the written answers. The PDF can be a scan of a *clearly handwritten* page, but typing the answers is *strongly* encouraged.

If not written otherwise, all the files should be at the root of the zip (meaning that they are not inside an inner folder).

The zip file should be of type .zip and not .rar.

Unless stated otherwise, all assignments must be written in Python 3.8.

We set up a docker environment with python3 all required packages to allow you to run and test your code. You can use other developing environments for writing your code, but you need to make sure it runs on the python version installed in the docker before submitting it.

Code that fails to run inside the docker environment will not be graded!

Instruction for running the python3 environment:

1. Login to nova or any cs server of your choice.
2. Run the following 2 commands to start the docker:
   ```
   export UDOCKER_DIR="/specific/netapp5_2/eyalron1/SecCourseDocker"
   udocker run --bindhome SecDock
   ```
3. If all works well, you should now be running inside the docker, with the docker's home directory mapped to your own home directory. Note that

you can only save files inside your home directory.

4. You can run python with the command:
`python3`

2 Coding Assignment 4

For your fourth coding assignment, you are requested to implement two attacks.

2.1 Lucky 13 Attack on CBC-HMAC

The first is the lucky 13 plaintext recovery attack on CBC-HMAC. For the purposes of this attack, you are provided with an oracle that given a ciphertext c (and an IV) tells how many calls were made to the compression function of the HMAC.

Given a ciphertext c and an index t, the attack should return a list of candidates for the lower two bytes of the t'th block of the encrypted plaintext.

For this attack, please implement the missing parts (denoted by "?") in the file `lucky_13.py`. To help you check your work, a file with test vectors is included.

2.2 Bellcore fault attack on RSA

The second attack is the Bellcore fault attack on RSA. For the purposes of this attack, you are provided with an oracle that partially decrypts RSA by calculating

$$c^d \, mod \, p$$

and

$$c^d \, mod \, q$$

You must:

1. Implement the oracle `RSA_oracle`, that completes the decryption process using the Chinese Remainder Theorem.
2. Implement the function `bellcore_attack`, that utilizes `RSA_oracle` to factor the public modulus n.

For this attack, please implement the missing parts (denoted by "?") in the file `bellcore.py`. To help you check your work, the function returns `None` if the recovered p and q are incorrect.