Homework Assignment 1
Information Security - Theory Vs. Reality
0368-4474
Submit by: 30th of April 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. To submit your assignment, please prepare a zip file of the format `HW_x_id`, where x is the number of the assignment and id is your id number.

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 1

For your first coding assignment, you are requested to implement two cryptographic algorithms.

The first one is an authenticated encryption with associated data (AEAD) encryption mode of operation based on CBC encryption and HMAC for authentication, and is usually referred to as CBC HMAC. The decryption of this mode can be found in the attached "CBC-HMAC Documentation" file. After reading this file, you should implement the missing parts (donated by ?) in the file "CBC-HMAC.py". Note that the decryption process should validate the correctness of both the MAC and the padding. If the validation fails, the function should return `None`.

The second algorithm is an RSA-based public-key encryption and decryption algorithm scheme called PKCS 1.5. Its full description can be found in the attached public RFC 2313. Again, after reading "rfc2313.txt", you should implement the missing parts (donated by ?) in the file "PKCS-1.5.py". Note that the decryption process should validate the correctness of the padding. If the validation fails, the function should return `None`.

To help you test your implementation's correctness, we also attach test vectors that you can use as input to your encryption function and compare the resulting ciphertext.

3 Question 1

After you finish implementing both algorithms, answer the following questions for each algorithm:

Similar to what was described in class, we assume an attacker is able to measure the exact run time of the decryption function. What if any information can such an attacker learn about decrypted valid messages? What if any information can such an attacker learn about invalid decrypted messages (e.g., messages with invalid padding)?