STORING PASSWORDS

Deliverable for this lab: briefly add the following information to the deliverable

- ***1.1: screenshot of the Bitwarden dashboard after you log in***
- ***1.2: screenshot of Burp Suite interception window – feel free to grey out your password if it is visible***
- ***1.2: short explanation why the password is (not) visible***
- ***1.3: screenshot of your memory consumption during the calculation of the Argon2id hash calculation.***
- ***1.3: screenshot of your Python code.***

1.1 Password manager

Start using a password manager today.

In this exercise, we will use Bitwarden as password manager. If you already use another password manager, great! However, please do create an account at Bitwarden for the sake of this exercise. You can do so at https://vault.bitwarden.com/#/register .

1.2 Intercept calls to Bitwarden Vault

Now start Burp and intercept the calls that are sent to Bitwarden's servers during login. Can you see your password? Why (not)?

Hint: did you read https://bitwarden.com/images/resources/security-white-paper-download.pdf ?

1.3 Build a simple password storage mechanism in Python

Build a Python console application that uses the pynacl library and that has the following functionalities:

1. it takes a plaintext as input and that prints the Argon2id hash representation (storage) to console;
2. it takes two inputs: a) a plaintext input and b) an Argon2id hash and that verifies that the hash of the plaintext input matches the Argon2id hash (verification).

When doing this, try to max out the RAM of your machine by playing with the options of the pynacl Argon2id function.