Report: Werkzeug

Note: There are some references to line numbers, these refer to line numbers to the linked file that can be accessed by the link provided above.

The project utilizes the werkzeug.security library, the code for which can be found at: https://github.com/pallets/werkzeug/blob/master/src/werkzeug/security.py

More specifically the project utilizes two specific functions, from this library, `generate_password_hash` & `check_password_hash`. A detailed summary of how the library generates and checks password hashes can be found in this document.

- `generate_password_hash (Line#153)`

This function accepts a plain text password string and outputs a password hash. The format of the hashed string is `method$salt$hash`.

The method by default is "`pbkdf2:sha256`". The actual method that comprises of the first part of the three part output is "`pbkdf2:method:iterations`".

The salt is a 16 character long string comprising of random characters, which are chosen from the set {{a-z}, {A-Z}, {0-9}}. The library further utilizes the secrets library to pick characters for the salt.

All string passwords are first utf-8 encoded. The hashlib library is further utilized to compute the hash, more specifically, the `pbkdf2_hmac` method of the hashlib is used to derive the hash. The `pbkdf2_hmac` method uses bitwise manipulation to compute the hash, the data is xor'ed for a set amount of iterations.

- `check_password_hash (Line#185)`

This function accepts a plain text password and a hash generated by `generate_password_hash` and returns a boolean value.

This method uses the same methodology as the above function, it computes the hash of the given plain text password, and compares it to the hash provided in the `method$salt$hash` generated output by `generate_password_hash` function.

This method encodes both the plain text and hash strings to utf-8. It further utilizes the hmac library to compare the two byte strings.

The project utilizes the werkzeug.utils library, the code for which can be found at: https://github.com/pallets/werkzeug/blob/master/src/werkzeug/utils.py

- `secure_filename (Line#430)`

The secure filename functions cleans the passed string in various ways to ensure that the passed filename will not cause any kind of html injection or other security issues.

The function first normalizes the string through unicode normalization, and then encodes it back into ascii. From there, it replaces any possibly troubling characters with empty spaces. Last, it applies a couple of windows specific filters to ensure that the filename is not anything that windows would deem a special name.

This function does not utilize any other special libraries, and simply acts as a smart and easy way for the user to ensure the security of their filename.

Licensing:

This is a modified verison of the BSD-3 Clause, which allows it to be used commercially and privately, and you are free to modify and redistribute it as well. The only thing it limits is in terms of liability and warranty on the creators part. Basically, like they say, the software is provided ‘as-is’, and we are free to use it however we please, but any faults or otherwise from this code does not fall on its creators.