Sensitive Data Exposure Prevention in PHP

1. If web applications don't protect sensitive data such as financial information and passwords, attackers can gain access to these data and sell or utilize it for nefarious purposes. One popular method for stealing sensitive information is using a man-in-the-middle attack.

2. Data exposure risk can be minimized by encrypting all sensitive data as well as disabling the caching of any sensitive information. Additionally, web application developers should take care to ensure that they are not unnecessarily storing any sensitive data.

Prevention:

1. Checking type of password hash algorithms inside database:
    a. There are plenty of hash algorithms in the security world, however there are some that are weak and outdated and shouldn't be used.

    b. The toolkit will check some types of hash algorithms such as: MD5, SHA-1, SHA-256, SHA-384, SHA-512, BCRYPT.

    c. it will use hash length as the signature of the hash algorithm
        i. MD5 16 bytes hash algorithm = 32 chars
        ii. SHA-1 20 bytes hash algorithm = 40 chars
        iii. BCRYPT 30 bytes salt+hash algorithm = 60 chars
        iv. SHA-256 32 bytes hash algorithm = 64 chars
        v. SHA-384 48 bytes hash algorithm = 96 chars
        vi. SHA-512 64 bytes hash algorithm = 128 chars

    d. Notes:
        i. the longer char length the hash algorithm outputs, the more it is secured.
        ii. some other algorithms have the same char length, but they have different specifications, but we will concentrate on the length here.
        iii. Special rule is applied, if password hash length is less than 24 chars, it won't be considered as hashed password.

    e. SALT must be applied, and SALT's length must be at least same of output hash, and must be Random for each password.

    f. If Password hash less than 60 chars, it won't be considered as secured.

    To read more about hashes, go here:
    https://www.php.net/manual/en/function.hash.php

2. Make sure HTTPS is used by force redirecting HTTP to HTTPS, using (.htaccess)
    a. First check the existence of the ".htaccess" in the project root
    b. Then write the following lines in it if not already written:

- RewriteEngine On
- RewriteCond %{SERVER_PORT} 80
- RewriteRule ^(.*)$ https://www.yourdomain.com/$1 [R,L]

c. Above lines of code forces clients to be redirected from HTTP to HTTPS

To read more about htaccess redirection, go here:
https://www.freecodecamp.org/news/how-to-redirect-http-to-https-using-htaccess/

All Above Explained ideas will be Scanned in the Toolkit.

3. Some Verbal Advices

a. Use HTTPS if possible using Ephemeral Diffie-Helman Key Exchange not RSA to guarantee PFS (Perfect Forward Secrecy).
b. Implement HSTS (HTTP Strict Transport Security): to Force HTTPS on browsers and Hackers cannot override the first HTTP redirect to HTTPS
c. Encrypt sensitive data such as Credit Card info and SSN and NOT TO Store Encryption Keys with them.
d. Use Pepper next to the Salt while hashing a password

To read more about Sensitive Data Exposure:
https://www.youtube.com/watch?v=2RKbacrkUBU
https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html
https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A3-Sensitive_Data_Exposure