TDT4237 Exercise 3: Mitigation

General information

This exercise is the third exercise in the course TDT4237 – Software Security and Data Privacy, and the second concerning with the SecureHelp application. The exercise counts for 30/100 points to be able to take the exam with at least 70 points in the course. The report and code should be submitted by the deadline on 23rd March 23:59.

The overall task in the exercise is to mitigate a predefined set of 10 vulnerabilities found within the application.

Any questions regarding the exercise should preferably be asked in the course forum on Blackboard. Alternatively, emails can be sent to tdt4237@idi.ntnu.no.

The task

The vulnerabilities in the application are listed in the solutions guide published on Blackboard. This exercise requires a predefined set of 10 vulnerabilities to be mitigated. In the report, you should explain how you have mitigated each vulnerability and show the code differences. The code differences should be provided as commits or branches from Gitlab, while the report will only include crucial code snippets. The vulnerabilities that should be fixed are:

1. WSTG-ATHN-01 / WSTG-CRYP-03 Sensitive Information Sent via Unencrypted Channels(**TLS**) .
2. WSTG-ATHN-03 Unlimited login attempts, no lockout.
3. WSTG-SESS-01 No timeout on email verification.
4. WSTG-SESS-01 Weak email verification link.
5. WSTG-ATHZ-02 Approve certification as a normal user.
6. WSTG-SESS-06 Access token only deleted client side.
7. WSTG-INPV-02 WSTG-CLNT-03 WSTG-CLNT-05 Unsanitized html field allowing injection.
8. WSTG-INPV-05 SQL injection when finishing a help request.
9. WSTG-CRYP-04 Insecure password hasher .
10. WSTG-CONF-12: CSP Default source not set.

Delivering code

Every group have been assigned to a repository on Gitlab with the following URL: https://gitlab.stud.idi.ntnu.no/tdt4237/2023/group-XXX . Everyone should have received an invitation to their repository and can also add members themselves. If there are any issues with accessing your repository then you should send an email.

The code pushed to the **production** branch within the deadline will be regarded as delivered code. This branch triggers the ***pipeline*** as defined in .gitlab-ci.yml. This will build and run your code on molde.idi.ntnu.no (similar to docker-compose up --build). Your pipeline must run successfully to assess your solution, it is important that the app still works after code changes. The report should only contain crucial code snippets. Otherwise, refer to relevant commits or branches.

Report and evaluation

You should create a report based on the template provided in blackboard. The maximum achievable points are 30, the content of the report and point distribution is as follows:

1. Mitigation of vulnerabilities (3 points for each vulnerability):
    a. Mitigation strategy (1.5)
    - The rationale to fix the vulnerability is correct.
    Explain why your proposed solution solves the problem.
    b. Code changes (1.5)
    - The code successfully mitigates the vulnerability.
    *Explain how your code works. The report can contain **CODE SNIPPETS**, otherwise, refer to **COMMITS, BRANCHES,** or **MERGE REQUESTS**.*

The report should be submitted to Blackboard within the deadline.

Presentation

The report will be evaluated by its presentation. Failure to meet the following requirements might lead to a reduction of the score, up to 1 point. Points are rarely deducted on this post, but we do need some minimal formal requirements on the content of the report. Those are:

- Language in a proper and academic style
- Reference list formatted according to the IEEE citation style
- The precision of facts
- A logical structure of content
- Plagiarism will not be tolerated

Approaching TLS

We inspect the changes required to implement TLS as it requires an assumption in SecureHelp.
*WSTG-ATHN-01 / WSTG-CRYP-03 Sensitive Information Sent via Unencrypted Channels(**TLS**).*
This vulnerability requires **one specific change** in the .env file before certificates can be used. The change is to set **PROTOCOL** value to *https*. Furthermore, your browser will not recognise your certificates and issue a warning.

```
GROUP_ID=250
PORT_PREFIX=21
DOMAIN=localhost
PRODUCTION=True
#PROTOCOL=http
PROTOCOL=https

DJANGO_SUPERUSER_PASSWORD=password1
DJANGO_SUPERUSER_USERNAME=admin1
DJANGO_SUPERUSER_EMAIL=admin@mail.com
```

The purpose of the PROTOCOL .env variable is to ensure the following:

- Backend: backend/securehelp/settings.py

    ```
    URL = PROTOCOL + '://' + DOMAIN + ':' + PORT_PREFIX + GROUP_ID
    ```

    Creates the URL to REDIRECT the user.
- Frontend: frontend/Dockerfile

```
RUN echo
"REACT_APP_API_URL='${PROTOCOL}://${DOMAIN}:${PORT_PREFIX}${GROUP_ID}/api'"
> ./.env
```

Creates the URL to reach the API in the React app.

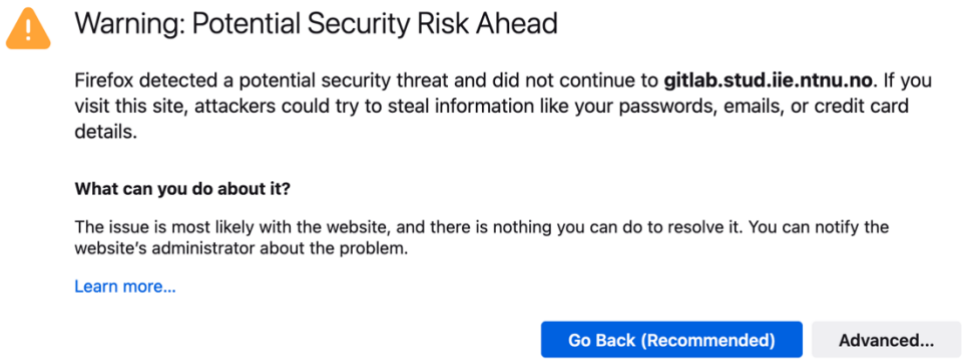
The documentation to configure NGINX can be found here:
https://nginx.org/en/docs/http/configuring_https_servers.html
You can change the *listen* value from 80 to 443 as the documentation suggests. However, port 80 will also work.

```
server {
    listen                  80 ssl;
```

It is easier to set the value to port 80 because it doesn't require further change for our purpose. If you choose to set the value to 443, then you also must change the value in docker-compose.yml:

```
45  ports:
46   - ${PORT_PREFIX}${GROUP_ID}:443
```

One final important thing is to note that the certificates that you need to generate will be *self-signed*. This means that **the browser will not recognize your certificate and issues a warning**.



This is **acceptable** for the purpose of this exercise and we accept self signed certificates. The warning can be bypassed by pressing "advanced".

~~

The other vulnerabilities does not require such specific assumption depending on other environment variables or code other than the source of the vulnerability.