Tyler Dao – Software Security

GitHub Repository: https://github.com/TylerDdao/Software-Security.git

Requirements

Functional Requirements

1. The app shall allow user to register for an account.
2. The app shall let the registered user to log in.
3. The app shall provide the user a way to log out
4. The app shall contain a user's session, so they don't have to log in with every action.
5. The app shall allow authenticated user to create notes.
6. The app shall allow authenticated user to view their notes.
7. The app shall allow authenticated user to modify their notes.
8. The app shall allow authenticated user to delete their notes.

Non-functional Requirements

1. The app shall not store any password in plain text.
2. The app shall encrypt all the notes when sending them between client and server.
3. The app shall have a secured, not easy to guess session ID.
4. The app shall prevent any SQL injection threats from the user.
5. The app shall prevent any invalid input from the user.
6. The app shall responses to any request in under 3 seconds on the normal internet speed.

Application Assets

Client's data: Credentials, note content, session information.

Server's data: Database, secret key.

Trust Boundaries

User Input → Server: User input is untrusted source; it must be sanitized before sending to the server to process.

Server → Client: The server should not send any client's sensitive information in the responses.

Server → Database: The server should not save plain text of sensitive information like password, note content received from the client to database.

Potential Threats

Main Features	Registration	Log In	Note Management
Spoofing	Not applicable because this creates a new account.	The attacker could guess the password and log in to impersonate an authorized user.	The attacker can steal a cookie session and see all the notes that belong to the authorized user of that session ID.
Tampering	The attacker could insert SQL queries into registration form to modify the database.	- The attacker could insert SQL queries into registration form to modify the database. - The attacker could modify session ID to impersonate another user.	The attacker could insert SQL queries in the note content to modify the database.
Repudiation	Not applicable because this is one-time process.	The sever can't know if the action is performed by the real user or the attacker.	The sever can't know if the action is performed by the real user or the attacker.
Information Disclosure	Password sent between client and server could be spied.	- Password sent between client and server could be spied. - If log in success, session ID could be spies if not properly handled.	Note content could be spies if not encrypted when sending between client and server.
Denial of Service	The attacker could send multiple requests to flood the server.	The attacker could send multiple requests to flood the server.	The attacker could send multiple requests to flood the server.
Elevation of Privilege	Not applicable because this feature doesn't have any access that requires special privilege.	Not applicable because this feature doesn't have any access that requires special privilege.	Once the attacker can impersonate an authorized user, they could perform actions that are out of their privilege.

Threats Ranking

After modeling the threats using STRIDE, I have summarized 4 main threats that could be performed on the Note App system.

- Attacker impersonates an authorized user.
- Attacker injects SQL queries in input.
- Attacker injects malicious script in input.
- Attacker steals session ID.
- Attacker floods the server with requests.

Threats	**Damage Potential**	**Reproducibility**	**Exploitability**	**Affected Users**	**Discoverability**	**Average Score**
Attacker impersonates an authorized user.	10	10	7	1	5	6.6
Attacker injects SQL queries in input	10	10	8	10 (All)	8	9.2
Attacker injects malicious script in input.	10	10	8	10 (All)	8	9.2
Attacker steals session ID	10	5	5	1	5	5.2
Attacker floods the server with requests.	6	10	10	10 (All)	10	9.2

Secure coding practice used

- The note content is encrypted by the server before saving to database or decrypted before sending to client.
- Rate limit is applied on the server, only allow maximum 500 requests/day/IP address or 50 requests/hour/IP address.
- The queries are parameterized to avoid injection attack.
- Note content is checked before displaying to prevent from malicious script.
- Login attempt is limited to maximum 5 attempts, after the maximum number of attempts reached, the user will not be able to login again in 15 minutes to prevent brute force attack.

Defense-in-depth strategies

- Database protection: Parameterized queries
- Application: Prevent application from active script in user's input
- Server-side protection: Apply rate limit for all API endpoints
- Client-side protection: User's password is hashed before sending to serverTest Report

Test Report

Test case	Purpose	Expected Result	Result
test_encrypt	To test the encrypt function	The encrypted data shall be different from plain text.	Pass
test_decrypt	To test the decrypt function	After encrypting plain text and then decrypting it, the plain text shall stay the same.	Pass
test_home_page	To test if home page is rendered.	There shall be string "Welcome to Note App" in response data and response status code == 200	Pass
test_login_page	To test if login page is rendered.	There shall be string "Login" in response data and response status code == 200	Pass
test_register_page	To test if register page is rendered.	There shall be string "Register" in response data and response status code == 200	Pass
test_notes_page_1	To test if notes page redirect user to login page if the user is not logged in.	There shall be string "Redirecting" in response data and response status code == 302	Pass
test_note_page_2	To test if notes page is rendered after the user is logged in.	There shall be string "Your Notes" in response data and response status code == 200	Pass
test_register_user	To test the register function in happy path.	An alert "User registered" shall be display in browser and response status code == 200	Pass
test_register_using_existing_username	To test if the user is able to register with the same username.	String "already registered" shall be include in an alert displayed in browser and response status code == 400	Pass
test_login_success	To test the login function in happy path.	String "Logged in" shall be include in an alert displayed in browser and response status code == 200	Pass
test_login_fail_incorrect_password	To test if user is able to log in with incorrect password	String "incorrect" shall be include in an alert displayed in browser and response status code == 401	Pass
test_login_with_unregistered_user	To test if user is able to log in with an unregistered username	String "not registered" shall be include in an alert displayed in browser and response status code == 401	Pass
test_lock_after_max_attempts	To test if the account is locked if user reached their max login attempts	String "max attempts" shall be include in an alert displayed in browser	Pass

test_logout	To test if the user is able to log out	String "Logged out" shall be include in an alert displayed in browser and response status code == 200	Pass
test_note_is_encrypted	To test if the note content is encrypted in the database	The note content retrieved from database shall be different from the plain text.	Pass
test_password_is_hashed	To test if user's password is hashed in the database	The password retrieved from database shall be different from the plain password.	Pass
test_add_note	To test if the user is able to add new note	An alert "Note saved" shall be display in browser and response status code == 200	Pass
test_get_note	To test if the notes are fetched	An alert "Note saved" shall be display in browser, note content shall be include in response data and response status code == 200	Pass
test_delete_note	To test if a note can be deleted	An alert "Note deleted" shall be display in browser and response status code == 200	Pass
test_modify_note	To test if a note can be modified	An alert "Note modified" shall be display in browser, new note content shall be include in response data and response status code == 200	Pass

Running instruction

- A README.md with instruction to build and run docker container, and instruction to run unit test.

References

Chou, E., & Groves, R. (2018). *Distributed Denial of Service (DDoS): practical detection and defense* (First edition.). O'Reilly Media.

Defence in depth and how it applies to web applications. (n.d.). https://www.acunetix.com/websitesecurity/defence-in-depth-and-how-it-applies-to-web-applications/.

Pauli, J. J. (2013). *The basics of web hacking: tools and techniques to attack the Web* (S. White, Ed.; 1st edition). Syngress, an imprint of Elsevier.