Started on	Monday, 2 June 2025, 4:39 PM
State	Finished
Completed on	Monday, 2 June 2025, 4:48 PM
Time taken	8 mins 48 secs
Marks	14.00/16.00
Grade	**87.50** out of 100.00

Question 1

Complete

Mark 1.00 out of 1.00

How can you prevent JWT replay attacks in sensitive RBAC-based applications?

- a. Implement rotating refresh tokens
- b. Use only the frontend to validate roles
- c. Store tokens in localStorage
- d. Use longer expiration time

Question 2

Complete

Mark 1.00 out of 1.00

If a user's role is updated from "editor" to "admin", but their JWT hasn't expired yet, what is a potential risk?

- a. Signature gets mismatched
- b. Token size increases
- c. Role update may not reflect until re-login
- d. Token becomes invalid immediately

Question 3

Complete

Mark 0.00 out of 1.00

In a RBAC model, which principle is crucial for minimizing access privileges?

- a. Least privilege
- b. Time-based access
- c. Token obfuscation
- d. Role inheritance

Question 4

Complete

Mark 1.00 out of 1.00

In a secure RBAC system, where should the logic for role-based route protection ideally reside?

- a. JWT header
- b. Middleware or backend route handlers
- c. Database triggers
- d. Frontend only

Question 5

Complete

Mark 1.00 out of 1.00

What change should be made to the following JWT-based login handler to add RBAC? const token = jwt.sign({ id: user.id }, 'mysecret');

- a. Add role: user.role to payload
- b. Encrypt the token
- c. Use HS512 algorithm
- d. Add user email to the payload

Question 6

Complete

Mark 1.00 out of 1.00

What is a secure way to refresh a short-lived JWT without asking the user to log in again?

- a. Use the same JWT for 1 year
- b. Store token in sessionStorage
- c. Use a secure refresh token mechanism
- d. Use a cookie-stored access token

Question 7

Complete

Mark 1.00 out of 1.00

What is the primary purpose of the JWT signature?

- a. Encrypts the token data
- b. Stores expiration timestamp
- c. Validates the integrity and authenticity of the token
- d. Prevents cross-site scripting attacks

Question 8

Complete

Mark 1.00 out of 1.00

What is the problem with the following code if used in production? const token = jwt.sign({ userId: 1 }, '123', { expiresIn: '2h' });

- a. Nothing, it's secure
- b. Token will never expire
- c. The secret is weak and predictable
- d. It uses numeric user ID

Question 9

Complete

Mark 1.00 out of 1.00

What will happen if the secret key used to sign a JWT is leaked?

- a. Token will become unreadable
- b. JWTs will auto-expire
- c. Signature verification will be stricter
- d. Any user can generate valid tokens

Question 10

Complete

Mark 1.00 out of 1.00

Which claim in a JWT helps enforce token expiration?

- a. aud
- b. exp
- c. iat
- d. sub

Question 11

Complete

Mark 1.00 out of 1.00

Which part of a JWT is typically used to store user roles for implementing RBAC?

- a. Payload
- b. Signature
- c. Token Expiry
- d. Header

Question 12

Complete

Mark 1.00 out of 1.00

Why is storing a JWT in localStorage considered risky in web applications?

- a. It expires too quickly
- b. It cannot be read by JavaScript
- c. It's vulnerable to XSS attacks
- d. It increases backend load

Question 13

Complete

Mark 0.00 out of 1.00

Given the following code, which statement is true?

```
const MyComponent = React.memo(({ onClick }) => {
  console.log("Rendered");
  return <button onClick={onClick}>Click</button>;
});
```

What must be true for React.memo to prevent re-renders when parent re-renders?

- a. onClick must be memoized using useMemo
- b. React.memo always skips rendering regardless of prop types
- c. onClick must be stable across renders (e.g., memoized using useCallback)
- d. onClick must be declared outside the parent component

Question 14

Complete

Mark 1.00 out of 1.00

In which of the following scenarios is useMemo most beneficial?

- a. To optimize expensive computations based on stable inputs
- b. To prevent unnecessary re-renders of pure components
- c. To memoize functions used as event handlers
- d. To store global constants across modules

Question 15

Complete

Mark 1.00 out of 1.00

Consider the following component:

```
const List = React.memo(({ items }) => {
  return items.map(item => <div key={item.id}>{item.name}</div>);
});
```

If the parent re-renders but passes the same array reference for items, what happens?

a. React.memo skips rendering because the array reference is unchanged

b. React.memo deep compares array values

c. React.memo skips rendering only if keys are stable

d. React.memo causes List to re-render

Question 16

Complete

Mark 1.00 out of 1.00

Why might excessive use of useMemo lead to performance degradation rather than improvement?

a. Creating memoized values and comparing dependencies has computational cost

b. React re-renders the component regardless of useMemo

c. useMemo increases memory usage permanently

d. useMemo causes stale closures