Started on	Monday, 2 June 2025, 4:39 PM
State	Finished
Completed on	Monday, 2 June 2025, 4:46 PM
Time taken	7 mins 41 secs
Marks	15.00/16.00
Grade	**93.75** 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. Role update may not reflect until re-login
- b. Token size increases
- c. Signature gets mismatched
- d. Token becomes invalid immediately

Question 3

Complete

Mark 1.00 out of 1.00

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

- a. Least privilege
- b. Token obfuscation
- c. Time-based access
- 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. Frontend only
- b. Middleware or backend route handlers
- c. JWT header
- d. Database triggers

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. Add user email to the payload
- c. Encrypt the token
- d. Use HS512 algorithm

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

Question 7

Complete

Mark 1.00 out of 1.00

What is the primary purpose of the JWT signature?

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

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

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. Signature verification will be stricter
- c. JWTs will auto-expire
- 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. sub
- b. aud
- c. exp
- d. iat

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

Question 12

Complete

Mark 1.00 out of 1.00

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

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

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

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 store global constants across modules
- c. To memoize functions used as event handlers
- d. To prevent unnecessary re-renders of pure components

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 deep compares array values
- b. React.memo skips rendering only if keys are stable
- c. React.memo causes List to re-render
- d. React.memo skips rendering because the array reference is unchanged

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 causes stale closures
- d. useMemo increases memory usage permanently